UNIVERZITA KARLOVA

PEDAGOGICKÁ FAKULTA

KATEDRA INFORMAČNÍCH TECHNOLOGIÍ A TECHNICKÉ VÝCHOVY

DIPLOMOVÁ PRÁCE

ROZVOJ ALGORITMICKÉHO MYŠLENÍ ŽÁKŮ ZŠ VE VÝUCE INFORMATICKY ZAMĚŘENÝCH PŘEDMĚTŮ S VYUŽITÍM SCRATCH

THE DEVELOPMENT OF ALGORITHMIC THINKING IN ELEMENTARY SCHOOL PUPILS USING THE SCRATCH PROGRAMMING TOOL

Bc. Milan Svoboda

Vedoucí diplomové práce: doc. RNDr. Miroslava Černochová, CSc.

Studijní program: Učitelství pro střední školy (N7504)

Učitelství VVP pro ZŠ a SŠ – informační a komunikační technologie

2017-2018

UNIVERZITA KARLOVA
PEDAGOGICKÁ FAKULTA
Katedra informačních technologií a technické výchovy

ZADÁNÍ DIPLOMOVÉ PRÁCE
akademický rok 2017/2018

Jméno a příjmení studenta: **Bc. Milan SVOBODA**

Studijní program: **N7504 Učitelství pro střední školy**

Studijní obor: **Učitelství VVP pro ZŠ a SŠ – informační a komunikační technologie**

Název tématu práce v českém jazyce:

Rozvoj algoritmického myšlení žáků ZŠ ve výuce informaticky zaměřených předmětů s využitím SCRATCH

Název tématu práce v anglickém jazyce:

The development of algorithmic thinking in elementary school pupils using the SCRATCH programming tool

Jazyk práce: **český jazyk**

Stručná charakteristika tématu:
Cílem práce je zjistit, do jaké míry aktivity se SCRATCH přispívají k rozvoji algoritmického myšlení žáků ZŠ.

Zásady pro vypracování:
- Seznamte se s vymezením konceptu algoritmického myšlení žáků.
- Navrhněte testové úlohy k ověření dopadu aktivit SCRATCH na utváření dovedností žáků algoritmicky myslet.
- Uspořádejte pedagogický experiment, jehož základem je výuka SCRATCH zaměřená na rozvoj algoritmického myšlení žáků 5. a 6. ročníku ZŠ.
- Formulujte závěry a doporučení pro vzdělávací praxi.

Předpokládaná struktura práce:
- Úvod – Vymezení výzkumného pole - Cíle a metodika práce - Teoretická východiska práce – Experimentální část - Výsledky práce - Závěr - Seznam použitých informačních zdrojů - Přílohy

Seznam doporučené literatury:

Při řešení budou použity materiály didaktiků informatiky v ČR, SR, zahraničních odborníků, weby soutěže informatický Bobřík /Beabras, sborníky konference DIDINFO, IFIP, ISSEP, EUROlogo aj.

Vedoucí diplomové práce: **Doc. RNDr. Miroslava Černochová, CSc.**

Oponent diplomové práce: PhDr. Petra Vaňková, Ph.D.

Předpokládaný rozsah diplomové práce[1]: **65 normostran, přílohy**

Datum zadání práce: **22. září 2017**

Předběžný termín odevzdání práce: **20. května 2018**

V Praze dne: 22. září 2017

.....
PhDr. Jiří Štípek, Ph.D.
vedoucí katedry

[1] Minimální rozsah bakalářské práce činí standardně 40 normostran (72 000 znaků vč. mezer) vlastního textu.

Odevzdáním této diplomové práce na téma Rozvoj algoritmického myšlení žáků ZŠ ve výuce informaticky zaměřených předmětů s využitím Scratch potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Datum: 12. 7. 2018 .....

Podpis

Poděkování

Rád bych poděkoval doc. RNDr. Miroslavě Černochové, CSc. za ochotu a podporu, kterou mi jako vedoucí mé diplomové práce při jejím zpracování poskytla.

NÁZEV:

Rozvoj algoritmického myšlení žáků ZŠ ve výuce informaticky zaměřených předmětů s využitím SCRATCH

AUTOR:

Bc. Milan Svoboda

KATEDRA:

Katedra informačních technologií a technické výchovy

VEDOUCÍ PRÁCE:

Doc. RNDr. Miroslava Černochová, CSc.

ABSTRAKT:

Diplomová práce se zabývá problematikou algoritmizace a programování na základních školách. Zkoumá vliv výuky základů programování ve Scratch na rozvoj algoritmického myšlení žáků a jejich schopnost rozvíjet logické myšlení a řešit problémové úlohy. Teoretická část se zabývá vymezením souvisejících pojmů a vztahem programování ke klíčovým kompetencím definovaným RVP ZV. Praktická část vyhodnocuje zkušenosti s výukou žáků 5. a 6. třídy základní školy v rámci pedagogického experimentu, jehož cílem bylo zkoumat vliv výuky s využitím vizuálního programovacího jazyka Scratch na rozvoj algoritmického myšlení žáků.

KLÍČOVÁ SLOVA:

Algoritmické myšlení, informatické myšlení, programování, Scratch

TITLE:

The development of algorithmic thinking in elementary school pupils using the SCRATCH programming tool

AUTHOR:

Bc. Milan Svoboda

DEPARTMENT:

Department of information technology and education

SUPERVISOR:

Doc. RNDr. Miroslava Černochová, CSc.

ABSTRACT:

The diploma thesis deals with problems of algorithmization and programming at elementary schools. It examines the influence of teaching the basics of programming in Scratch on the development of pupils' algorithmic thinking and their ability to develop logical thinking and problem solving. The theoretical part deals with the definition of related concepts and the relation of programming to the key competencies defined by the RVP ZV. The practical part evaluates the experience with the teaching of pupils of the 5th and 6th grades of elementary school within the pedagogical experiment, whose aim was to study the influence of teaching with the use of the visual programming language Scratch on the development of pupils' algorithmic thinking.

KEYWORDS:

Algorithmic thinking, informational thinking, programming, Scratch

Obsah

ÚVOD

Změn v oblasti digitálních technologií přibývá, jsou stále rychlejší a mají značný vliv na život a rozvoj společnosti. Překotný vývoj na poli vědy a techniky má za následek reorganizaci pracovního trhu a pro řadu lidí je stále obtížnější se těmto proměnám prostředí přizpůsobit. Adaptovat se na tuto transformaci společnosti by měly budoucím generacím napomáhat především školy, a proto se výuka v oblasti informačních technologií a informatiky stává stále důležitější. Bohužel, vzhledem k závratné rychlosti změn je nesnadné odhadnout, jakým směrem se bude společnost ubírat, a tedy i jaké znalosti a dovednosti budou žáci v budoucnu potřebovat. Z toho vyplývá, že i obsah výuky je velmi těžké předpovědět a stanovit. Je však jisté, že nelze naše žáky učit v oblasti digitálních technologií pouze uživatelským dovednostem. Je třeba vést žáky k porozumění obecným principům, ke kritickému myšlení a k řešení problémů. Je důležité zaměřit se na rozvoj samostatného, originálního, netradičního a tvořivého myšlení žáků, tedy na vlastnosti, které jsou pro uplatnění v současné informační společnosti naprosto nepostradatelné, a lze tedy předpokládat, že tomu tak bude i nadále. Jejich předpokladem jsou schopnost logického, abstraktního a nepochybně také algoritmického myšlení.

Cílem této práce je uspořádat a vyhodnotit pedagogický experiment, který se pokusí zodpovědět otázku, zda má výuka základů programování v jazyce Scratch pozitivní vliv na algoritmické myšlení žáků 5. a 6. tříd základní školy, zda přispívá k rozvoji jejich digitální gramotnosti a zda je tedy vhodné ji do vzdělávacích programů určených pro základní vzdělávání zařadit.

1 VYMEZENÍ VÝZKUMNÉHO PROBLÉMU

Diplomová práce se zabývá otázkou, zda aktivity s programovacím prostředím Scratch přispívají k rozvoji algoritmického myšlení žáků ZŠ. Zkoumá vliv výuky základů programování ve Scratch na algoritmické myšlení žáků a jejich schopnost řešit úlohy algoritmické povahy.

Teoretická část se zabývá vymezením souvisejících pojmů a vztahem programování ke klíčovým kompetencím definovaným v RVP ZV. Praktická část se věnuje pedagogickému experimentu, jehož cílem je zkoumat vliv výuky s využitím vizuálního programovacího jazyka Scratch na rozvoj algoritmického myšlení žáků.

Programování se v mnoha zemích na základních školách pěstovalo ve vzdělávání již v 60. letech 20. století především s využitím jazyka LOGO. Od té doby měly základní školy k dispozici různé edukační programovací jazyky, mezi něž se zařadil také Scratch. V souvislosti s kritikou stavu výuky informační výchovy a ICT vzdělávání v posledních letech požadavek na rozvíjení algoritmického myšlení žáků vzrůstá.

Rozvíjí výuka ve Scratch algoritmické myšlení žáků 5. a 6. ročníku ZŠ? Může výuka základů programování změnit způsob, jakým žáci řeší problémy? Je tvorba vlastních počítačových programů pro žáky motivujícím faktorem? Je vhodné zařadit programování ve Scratch do obsahu kurikulárních dokumentů? To jsou hlavní otázky, které byly východiskem pro vymezení výzkumného problému této diplomové práce.

2 CÍLE PRÁCE A POUŽITÉ METODY

Hlavním cílem práce je vymezit koncept algoritmického myšlení žáků a na jeho základě:

1) zjistit, do jaké míry aktivity se Scratch přispívají k rozvoji algoritmického myšlení žáků ZŠ;
2) navrhnout testové úlohy pro ověření dopadu aktivit se Scratch na utváření schopnosti žáků algoritmicky myslet;
3) navrhnout výuku základů programování pro žáky 5. a 6. třídy ZŠ ve Scratch;
4) uspořádat pedagogický experiment, jehož základem je výuka Scratch zaměřená na rozvoj algoritmického myšlení žáků 5. a 6. ročníku ZŠ.

2.1 Výzkumné otázky a úkoly

S ohledem na cíle práce byly stanoveny následující výzkumné otázky a s nimi přímo související hlavní úkoly a jejich součásti.

2.1.1 Výzkumné otázky

Otázka 1: Jaký mají žáci skupin zařazených do experimentu s využitím Scratch vztah k programování?

Z jakého prostředí pochází žáci, kteří se zúčastní experimentu? Mají již osvojeny základní dovednosti při používání počítače? Jaké množství času věnují práci na počítači a především při jakých činnostech? Je součástí těchto činností také programování ve Scratch nebo jiném programu určeném dětem? To jsou jen některé z otázek, které bylo nutné zodpovědět, aby bylo možné charakterizovat žáky na počátku pedagogického experimentu.

Otázka 2: Rozvíjí výuka ve Scratch u žáků 5. a 6. ročníku ZŠ algoritmické myšlení?

Hodinová dotace vzdělávací oblasti Informační a komunikační technologie v ČR je jedna hodina týdně na 1. i na 2. stupni ZŠ. Proto se žáci nejpozději ve věku 10 let seznamují ve školách s prací na počítači, učí se chápat funkci i důležitost výpočetní techniky a informací. Činnosti související s algoritmizací a programováním jsou však součástí jejich výuky v souladu se stávajícím RVP pouze výjimečně. Jaká je tedy úroveň algoritmického myšlení žáků ve věku 10–12 let, zůstává tedy otázkou. Pro účel této práce, sestavení charakteristiky skupin žáků a posouzení jejich vývoje v oblasti algoritmického myšlení v průběhu experimentu bylo nezbytné zjistit míru schopnosti žáků řešit úlohy algoritmické povahy na počátku experimentu, dříve než se žáci začnou seznamovat s programem Scratch.

Hlavním cílem pedagogického experimentu bylo uskutečnit šetření, které by ověřilo, zda výuka zaměřená na základy programování a aktivity se Scratch přispívá nejen k osvojení nových vědomostí a dovedností v oblasti programování, ale také k rozvoji algoritmického myšlení žáků a jejich kompetencí v oblasti řešení problémů. Může výuka programování s využitím vizuálního programovacího jazyka Scratch tyto kompetence rozvíjet? Aby bylo možno zodpovědět tuto otázku, bylo třeba ověřit rozdíl mezi těmito dovednostmi na počátku a na konci pedagogického experimentu. Nalezení odpovědi na tuto otázku může rovněž

pomoci při rozhodování, zda by se měla výuka programování začlenit do školního vzdělávání na základních školách.

Otázka 3: Zvýší výuka základů programování ve Scratch u žáků zájem o programování?

Motivace je hnacím motorem učení nejen u dětí na základní škole. Míra motivace tak přímo určuje chování žáků ve výuce. Bude tvorba vlastní počítačové hry v rámci závěrečného projektu faktorem, který podnítí úsilí žáků projekt dokončit? Budou se žáci věnovat činnostem spojeným s programováním i mimo školní prostředí? Budou pokračovat v práci se Scratch i po skončení výuky? Práce se pokusí zodpovědět i otázku, zda je možné pomocí výuky ve Scratch rozvinout u žáků zájem o programování.

Otázka 4: Je vhodné zařadit výuku programování ve Scratch do obsahu kurikulárních dokumentů?

Současné kurikulární dokumenty určené pro základní školy (RVP ZV, 2017) zmiňují algoritmické myšlení ve vzdělávacích oblastech Matematika a její aplikace a Informační a komunikační technologie. V případě vzdělávací oblasti ICT je algoritmizace zmíněna pouze v konkretizaci cílového zaměření vzdělávací oblasti Informační a komunikační technologie, nikoli však jako součást vymezení vzdělávacího obsahu. V návaznosti na data získaná v průběhu experimentu budou vyvozeny závěry, zda je vhodné zařadit výuku algoritmizace a programování do kurikulárních dokumentů ČR.

2.1.2 Výzkumné úkoly

Úkol 1: Zařadit výuku programování do výuky sledovaných skupin žáků 5. a 6. ročníku vybrané základní školy.

- Upravit příslušný ŠVP tak, aby umožňoval výuku základů programování v prostředí Scratch.
- Zajistit schválení upraveného ŠVP.

Úkol 2: Charakterizovat skupiny žáků, které se pedagogického experimentu účastní.

- Vytvořit dotazník, s jehož pomocí bude zjištěn vztah žáků k programování.

Úkol 3: Vytvořit sady vstupních a výstupních testů pro ověření úrovně algoritmického myšlení žáků.

- Ověřit přiměřenost obtížnosti úloh věkové kategorii žáků pomocí pilotního testování.
- Zdůvodnit algoritmickou podstatu navržených úloh.
- Zajistit srovnatelnou obtížnost vstupní a výstupní sady testů.

Úkol 4: Navrhnout výuku programování pro žáky 5. a 6. ročníku ZŠ v edukačním programovacím prostředí Scratch zaměřenou na programování vlastních her.

- Vytvořit metodické listy pro výuku jednotlivých lekcí.
- Vytvořit pro výuku pomocné soubory v Scratch.

Úkol 5: V rámci pravidelné výuky vzdělávací oblasti Informační a komunikační technologie realizovat s žáky 5. a 6. třídy ZŠ navrženou výuku základů programování s pomocí Scratch.

- Průběžně vyhodnocovat realizovanou výuku.
- V případě potřeby upravovat výuku tak, aby směřovala k dosažení co možná nejlepších výsledků v oblasti rozvoje myšlení žáků, získávání kompetencí v oblasti algoritmizace a schopnosti řešení problémů.

Úkol 6: Na závěr výuky realizovat se žáky závěrečný projekt v podobě tvorby počítačové hry v prostředí Scratch.

Úkol 7: Ověřit dopady pololetní výuky v prostředí Scratch

- Ověřit rozvoj schopnosti algoritmického myšlení žáků.
- Ověřit dopad výuky na vztah žáků k programování.

Úkol 8: Vyhodnotit pedagogický experiment. Stanovit závěry a doporučení pro pedagogickou praxi v oblasti výuky vzdělávacího oboru Informační a komunikační technologie na základní škole.

2.2 Metodologie výzkumu

K dosažení cílů diplomové práce byly při zjišťování informací použity kvalitativní i kvantitativní výzkumné metody.

První část praktické části představuje průzkum, jehož úkolem je charakterizovat skupiny žáků, kteří se pedagogického experimentu účastnili. Zdrojem pro získání informací v úvodní části průzkumu je dotazník (DOT 01, viz Příloha A). Ten zjišťuje množství času, který žáci stráví u počítače, způsoby jeho využití a vztah žáků k programování. Dotazník obsahuje 16 uzavřených a 9 otevřených otázek.

Ve druhé části průzkumu jsou ověřovány vstupní algoritmické schopnosti žáků pomocí sady testových úloh. Úvodní test (TEST A1 viz Příloha D) se skládá z celkového počtu 16 úloh, z toho 12 uzavřených a 4 otevřených. Dvě z úloh jsou rozděleny do dvou částí, které byly bodově hodnoceny samostatně. V testu lze získat maximálně 18 bodů.

Třetí část praktické části představuje pedagogický experiment, jehož hlavní složkou byla výuka základů programování v prostředí vizuálního programovacího jazyka Scratch dle předem připravených materiálů. V průběhu experimentu byly všechny skupiny žáků 5. a 6. ročníku, které se experimentu účastnily, vyučovány za obdobných, kontrolovaných podmínek s cílem minimalizovat rušivé vlivy vnějšího prostředí. Cílem experimentu bylo ověření hypotézy, že výuka pomocí Scratch může rozvíjet algoritmické myšlení žáků ZŠ.

Čtvrtou část výzkumu představoval časový úsek následující po výuce, v jehož průběhu žáci zapojení do experimentu samostatně navrhli a vytvořili vlastní počítačové hry v prostředí Scratch s využitím konceptů, které si osvojili v průběhu předchozí výuky.

V poslední části praktické části byla provedena analýza žákovských závěrečných projektů a v nich použitých konceptů za pomoci nástroje pro analýzu kódu Dr. Scratch. Poté byla provedena analýza dopadu výuky pomocí testu zjišťujícího schopnost porozumění kódu a algoritmizace (TEST B1, viz Příloha C) a kontrolní šetření výstupních algoritmických schopností žáků pomocí upravené sady testových úloh (TEST A2, viz Příloha E). Následoval dotazník (DOT 02, viz Příloha G) zjišťující vztah žáků k programování po ukončení pedagogického experimentu. Vyplnění dotazníků a testových úloh respondenty

bylo anonymní a dobrovolné. Všechny výsledky průzkumu byly vyhodnoceny, zpracovány graficky i formou tabulek a ze získaných dat byly vyvozeny příslušné závěry.

Tabulka 1 – Fáze pedagogického experimentu

	Přípravná fáze		Pracovní fáze (pedagogický experiment)		Dopady experimentu	
	Fáze 1	Fáze 2	Fáze 3	Fáze 4	Fáze 5	Fáze 6
Datum	25. 9. 2017 – 29. 9. 2017	2. 10. 2017 – 12. 10. 2017	16. 10. 2017 – 12. 2. 2018	12. 2. 2018 – 5. 4. 2018	9. 4. 2018 – 20. 4. 2018	21. 5. 2018 – 31. 5. 2018
Otázka	Kdo jsou žáci, kteří se účastní experimentu?	Jaké jsou algoritmické dovednosti žáků na počátku pedagogického experimentu?	Lze u žáků zařazených do experimentu vyučovat základy programování v prostředí Scratch?	Umožní předchozí výuka žákům samostatně programovat v prostředí Scratch?	Jaké jsou algoritmické dovednosti žáků po ukončení pedagogického experimentu?	Jaký je vztah žáků k programování po ukončení pedagogického experimentu?
Činnosti učitele a žáků	Definování vzorku žáků účastnících se experimentu.	Ověřování algoritmických dovedností žáků.	Výuka Scratch.	Tvorba závěrečných projektů.	Ověřování algoritmických dovedností žáků.	Ověřování dopadu výuky na vztah žáků k programování.
Metoda ověřování	DOT 01	TEST A1	TEST B1	Dr. SCRATCH	TEST A2	DOT 02

Tabulka 2 – Použité výzkumné nástroje

Fáze výzkumu	Popis	Vyuč. hod.
DOT 01	Dotazník pro zjišťování vztahu žáků k informačním technologiím, zaměřený na množství času, který žáci stráví u počítače, způsoby jeho využití a vztah žáků k programování. Dotazník obsahuje 16 uzavřených a 9 otevřených otázek.	1
TEST A1	Vstupní sada testových úloh pro zjištění algoritmických schopností žáků před pedagogickým experimentem. Test se skládá z celkového počtu 16 úloh, z toho 12 uzavřených a 4 otevřených. Dvě z úloh jsou rozděleny do dvou částí, které byly bodově hodnoceny samostatně. V testu lze získat maximálně 18 bodů.	1
Pedagogický experiment část 1.	Výuka základních programovacích konceptů pomocí vizuálního programovacího jazyka Scratch. Výuka sestává z 11 lekcí, vyučovaných v průběhu 14 hodin.	14
Pedagogický experiment část 2.	Závěrečný projekt – návrh a tvorba počítačové hry ve Scratch.	8
Dr. Scratch	Analýza úrovně použitých programovacích konceptů v závěrečných projektech za využití nástroje Dr. Scratch.	-
TEST B1	Test pro zjišťování dopadu výuky – schopnost algoritmizace a porozumění kódu. Test se skládá z celkového počtu 17 úloh, z toho je 12 úloh uzavřených a 5 úloh zaměřených na doplnění vývojových diagramů podle scriptů ve Scratch V testu lze získat maximálně 17 bodů.	1
TEST A2	Výstupní sada testových úloh pro zjištění dopadu výuky na algoritmické myšlení žáků po ukončení pedagogického experimentu. Test se skládá z celkového počtu 16 úloh, z toho 12 uzavřených a 4 otevřených. Dvě z úloh jsou rozděleny do dvou částí, které byly bodově hodnoceny samostatně. V testu lze získat maximálně 18 bodů.	1
DOT 02	Dotazník pro zjišťování dopadu výuky na vztah žáků k programování. Dotazník obsahuje 23 uzavřených a 3 otevřené otázky.	1

3 TEORETICKÁ ČÁST

3.1 Informatické myšlení

Termínem informatické myšlení je do českého jazyka překládán v zahraničí používaný termín computational thinking (CT). Poprvé tento pojem použil v roce 1996 americký matematik, informatik a profesor na Massachusettském technologickém institutu Seymour Papert. Uvedl jej v textu, zabývajícím se problematikou využití výpočetní techniky pro zavádění inovativních postupů při výuce matematiky. (Lessner, 2015)

Další, kdo koncept informatického myšlení více rozpracoval, je J. M. Wing z Columbia University. V článku Computing thinking z roku 2006 představila informatické myšlení jako univerzálně aplikovatelnou dovednost, která je určena všem, nejen odborníkům v oblasti výpočetní techniky. (Wing, 2006)

J. M. Wing (2006) argumentovala, že informatické myšlení zahrnuje tři klíčové části – 3A: algoritmy, abstrakci a automatizaci. Algoritmus, podobně jako recept, je řada pokynů krok za krokem. Abstrakce zahrnuje zobecnění a přenos procesu řešení problémů na obdobné problémy a automatizace zahrnuje mechanizaci řešení za použití digitálních nástrojů. J. M. Wing definuje informatické myšlení jako postupy použité při formulování problémů a jejich řešení. Součástí informatického myšlení by tedy dle J. M. Wing měly být tyto schopnosti:

- pochopit, které aspekty problému jsou řešitelné strojově,
- vyhodnotit souvislosti mezi informatickými prostředky a problémem,
- porozumět možnostem a omezením informatických prostředků,
- použít informatické prostředky novým způsobem či v nové situaci (nebo prostředky přizpůsobit),
- použít informatické strategie v jakékoliv oblasti. (Wing, 2010)

J. M. Wing vymezuje informatické myšlení jako základní schopnost stejně důležitou, jako je čtení, psaní a počítání. Nedomnívá se, že je třeba učit žáky informatickému myšlení, abychom vychovali pro trh práce více specialistů v oboru informatiky, ale pokouší se ukázat,

že dovednosti z oblasti informatiky lze uplatnit v mnoha dalších oborech a také při řešení každodenních problémů. (Wing, 2006.)

M. Sysło zdůrazňuje, že neexistuje shoda na definici tohoto pojmu, a to i proto, že koncept informatického myšlení (computational thinking, dále jen CT) je stále ještě poměrně nový. Janusz Krupa vyjádřil obavy ohledně důsledků zavedení termínu CT; přestože nové polské učební osnovy odkazují na CT, učitelé se tohoto pojmu „bojí“, a proto by bylo lepší používat známé pojmy jako řešení problémů, algoritmické myšlení a kritické myšlení. (Bocconi et al., 2016)

Řada autorů zahrnuje do CT širokou škálu dovedností:

- řešení problémů, analýza dat a zpochybňování faktů (Charlton & Luckin, 2012),
- shromažďování, analyzování a reprezentování dat, dekompozici problémů, používání algoritmů a postupů, vytváření simulací (Gretter & Yadav, 2016),
- simulace scénářů pomocí počítačových modelů (Creative Learning Exchange, 2015),
- dovednosti zabývající se otevřenými problémy a přetrvávajícími v náročných případech (Weintrop et al., 2015),
- úvahy o abstraktních objektech (Armoni, 2010).

Mezinárodní společnost pro technologie ve vzdělávání (ISTE) a Asociace učitelů informatiky (CSTA) zformulovaly definici informatického myšlení, která měla velmi pozitivní ohlasy u více než 700 učitelů informatiky a mnoha dalších odborníků z oblasti technologií. Vyplývá z ní, že informatické myšlení (CT) je proces řešení problémů, který zahrnuje (ale není omezen na) následující aktivity a dovednosti:

- *formulovat problémy způsobem, který umožňuje jejich strojové řešení*
- *logicky uspořádávat a zkoumat data*
- *reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace*
- *automatizovat řešení pomocí algoritmického myšlení (jako posloupnost kroků)*
- *odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů*

- *zobecňovat a přenášet tento postup řešení problémů do nejrůznějších dalších oblastí* (CSTA & ISTE, 2009)

Tyto dovednosti jsou podpořeny předpoklady a postoji, které jsou taktéž nezbytnou součástí CT:

- *sebejistota tváří v tvář složitosti*
- *vytrvalost při řešení obtížného problému*
- *snášení nejednoznačnosti*
- *schopnost vypořádat se s otevřenými problémy*
- *schopnost dorozumět se a spolupracovat s ostatními při dosahování společného cíle* (CSTA & ISTE, 2009)

Strategie digitálního vzdělávání schválená vládou v roce 2014 (MŠMT, 2014) zmiňuje CT jako sadu rozličných dovedností, které souvisejí s řešením problémů. Dle tvůrců se jedná o jeden z důležitých konceptů pro pochopení a participaci ve světě, který by bylo vhodné zavést do národního kurikula.

„Informatické myšlení zahrnuje jak dovednosti rozvíjené většinou vzdělávacích oborů (kreativitu, schopnost vysvětlování a týmové práce), tak i dovednosti řešení problémů, schopnost logického a algoritmického myšlení, schopnosti strukturace, abstrakce nad objekty a procesy, schopnosti vyvíjet technologie a porozumět tomu, jak fungují. V prostředí všudypřítomných digitálních technologií je základní pochopení jejich konceptů, dovednost je ovládat a modifikovat jejich funkce dle vlastních požadavků důležitým předpokladem pro jejich smysluplné a efektivní využívání.“ (MŠMT, 2014)

V zahraniční literatuře je pojem informatické myšlení někdy dokonce přímo zaměňován s termínem algoritmic thinking, tedy algoritmické myšlení. Například J. Gal-Ezer (v Bocconi et al., 2016) se přiklání k termínu algoritmické myšlení pro všechny dovednosti, které s počítači úzce souvisí.

„Dávám přednost použití výrazu Algoritmické myšlení pro druh dovedností, které lidé zahrnují do CT, protože pojem "computing" je spojován s výpočetní technikou, zatímco my děláme mnohem víc. Nejenom že počítáme, ale řešíme problémy tak, že procházíme několika úrovněmi abstrakce; na konci sice máme také výpočet, ale to je jen poslední fáze.

Algoritmické myšlení je duchem a uměním výpočetní techniky.“ (Gal-Ezer v Bocconi et al., 2016, str. 22)

Přes široké spektrum názorů, definic a návrhů se v literatuře stále opakuje soubor základních pojmů a dovedností spojovaných s informatickým myšlením. V tabulce č. 3 jsou dovednosti opakovaně zmiňované v pěti dokumentech, které jsou velmi často citovány, souhrnně informují o mnoha dalších studiích a obsahují názory z řady hledisek výzkumných oblastí a mezinárodních pracovních skupin (např. Pracovní skupina CSTA pro CT, výpočetní techniky ve škole, kurikulum pracovní skupiny IFIP a skupina Učebních osnov EDUsummIT TWG9). Dovednosti zmiňované v těchto klíčových dokumentech jsou navíc v souladu s dovednostmi, které se objevují ve zbývající části literatury revidované ve studii CompuThink. (Bocconi et al., 2016).

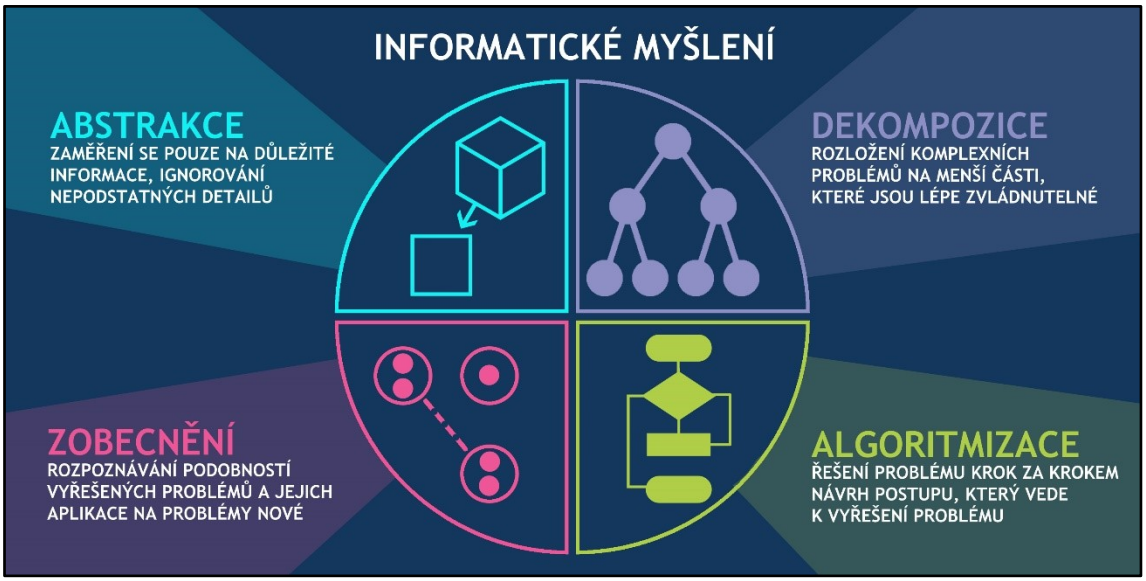
Tabulka 3 – Složky a dovednosti informatického myšlení v literatuře (převzato a přeloženo z Bocconi et al., 2016)

Barr & Stephenson 2011	Lee et al., 2011	Grover & Pea 2013	Selby & Woollard 2013	Angeli et al. 2016
Abstrakce	Abstrakce	Abstrakce a generalizace vzoru	Abstrakce	Abstrakce
Algoritmy a postupy		Algoritmické pojmy (řídící struktury)	Algoritmické myšlení	Algoritmy (včetně postupů a řídící struktury)
Automatizace	Automatizace			
	Analýza			
		Podmíněná logika		
Dekompozice problému		Strukturovaná dekompozice problému	Dekompozice	Dekompozice
		Ladění a systematické zjišťování chyb		Ladění
		Efektivita a omezení výkonu	Hodnocení	
			Generalizace	Generalizace
		Iterativní, rekurzivní a paralelní myšlení		
Paralelizace				
Simulace		Symbolové systémy a reprezentace		
		Systematické zpracování informací		

Přestože tedy stále chybí všeobecná shoda na definici informatického myšlení, v této oblasti se v literatuře opakovaně objevuje soubor základních pojmů a dovedností jako abstrakce, algoritmické myšlení, automatizace, rozklad, ladění a zobecňování. Z předchozích textů a tabulky č. 3 vyplývá, že nejčastěji je pojem CT spojován se základními dovednostmi, jako je právě abstrakce, zobecnění, dekompozice a algoritmizace.

Všechny čtyři pilíře informatického myšlení jsou stejně důležité (Wing, 2006):

- **Abstrakce** – zaměření se pouze na důležité informace, ignorování nepodstatných detailů.
- **Zobecnění** – rozpoznávání podobností vyřešených problémů a jejich aplikace na problémy nové.
- **Dekompozice** – rozložení komplexních problémů na menší části, které jsou lépe zvládnutelné.
- **Algoritmizace** – řešení problémů krok za krokem, nebo návrh postupu, který vede k vyřešení problému.



Obrázek 1 – Informatické myšlení (zdroj autor)

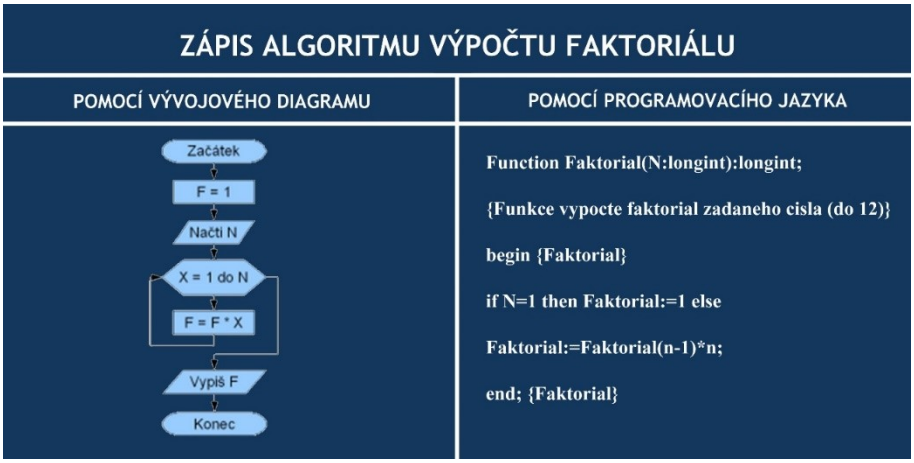
Algoritmizace je tedy jednou ze čtyř základních kompetencí utvářející informatické myšlení jedince a je chápána jako schopnost algoritmického myšlení. Jedná se tedy o: „*Proces převodu problému na jednotlivé kroky, který nazýváme algoritmizace.“* (Schubert, 2011, s. 67)

3.2 Algoritmické myšlení

Slovo „algoritmus“ je pro děti většinou zcela neznámým výrazem. Ve skutečnosti jsou však algoritmy všude kolem nás jako běžná součást našeho každodenního života. Řídí technologie, se kterými se všichni denně setkáváme. Telefony, automatické pračky, výtahy či semafory jsou jen některé příklady zařízení, která fungují na základě algoritmů. Algoritmy, i když se mohou jevit jako poměrně složité, mohou být ve skutečnosti docela prosté. Přestože si toho často nejsme vědomi, používáme velké množství postupů, které obsahují algoritmy v podobě rozhodování, řazení či opakování základních kroků. Například cesta do práce, vaření podle receptu, výměna rezervního kola u automobilu, montáž nábytku jsou situace, při kterých je nezbytné postupovat správně. Všechny tyto postupy jsou tvořeny řadou elementárních kroků, které, pokud jsou vykonávány v řádném pořadí, vedou k řešení konkrétního problému. V opačném případě můžeme v nejlepším případě ztratit „jen“ v dnešní době tak cenný čas.

„Algoritmus je jednoznačný srozumitelný přesně specifikovaný postup (předpis) vedoucí v konečném počtu kroků k řešení určité úlohy.“ (Havelková, 2008, str. 5)

Algoritmus je v podstatě posloupnost příkazů (pokynů či instrukcí), které po vykonání vedou k určitému cíli. Lze jej zapisovat buď slovně – běžným jazykem (kuchařské recepty, montážní návody), algoritmickým jazykem, programovacím jazykem, graficky (obrázkové návody, origami), nebo pomocí vývojových diagramů. Při zápisu algoritmu je třeba si uvědomit, kdo bude algoritmus vykonávat. V případě člověka stačí běžný slovní popis, pro počítač je třeba použít příslušný programovací jazyk.



Obrázek 2 – Příklady zápisu algoritmu (zdroj autor)

Všechny obecné postupy však nelze považovat za algoritmy. Jak se tedy liší soubor pokynů pro vykonání nějakého úkolu od toho, co lze nazvat algoritmem? Přestože je často slovem algoritmus označován jakýkoli obecný postup, který řeší zadaný úkol, měl by jím být nazýván pouze takový proces, který splňuje konkrétní, důležité požadavky. Těmi jsou především konečnost, obecnost, determinovanost, resultativnost a elementárnost.

- **Konečnost (finitnost)**: Splnění této vlastnosti má zabezpečit, aby algoritmus vždy skončil po vykonání konečného počtu kroků. Tento počet kroků může být libovolně velký (například šifrovací algoritmy mohou hledat řešení problému velmi dlouho), ale pro každý jednotlivý vstup musí být konečný. (Skalka et al., 2007)
- **Obecnost (hromadnost, masovost, univerzálnost): A**lgoritmus by měl být použitelný pro celou třídu obdobných problémů. Existují ale i algoritmy vytvořené pouze pro jeden účel, proto je tato vlastnost považována za užitečnou, nikoli za nutnou. (Skalka et al., 2007)

  Algoritmus by neměl řešit problém jak seřadit knihy v jedné určité knihovně, aby bylo možno co nejrychleji najít konkrétní knihu, ale jak řadit knihy obecně ve všech knihovnách, abychom dokázali co nejrychleji nalézt jakoukoli knihu.
- **Determinovanost**: Postup musí být sestavený tak, aby v každém momentě vykonávání bylo jednoznačně určené, jaká činnost má následovat, nebo zda algoritmus skončil. V případě stejných vstupů musíme po vykonání algoritmu obdržet i stejné výstupy. (Skalka et al., 2007)

- **Výstup (resultativnost):** Algoritmus má po konečném počtu kroků alespoň jeden výstup. Algoritmus musí řešit zadaný problém. (Skalka et al., 2007)
- **Elementárnost**: Postup je složený z jednoduchých (základních) kroků, které jsou pro vykonavatele (počítač, člověk) srozumitelné. (Skalka et al., 2007)

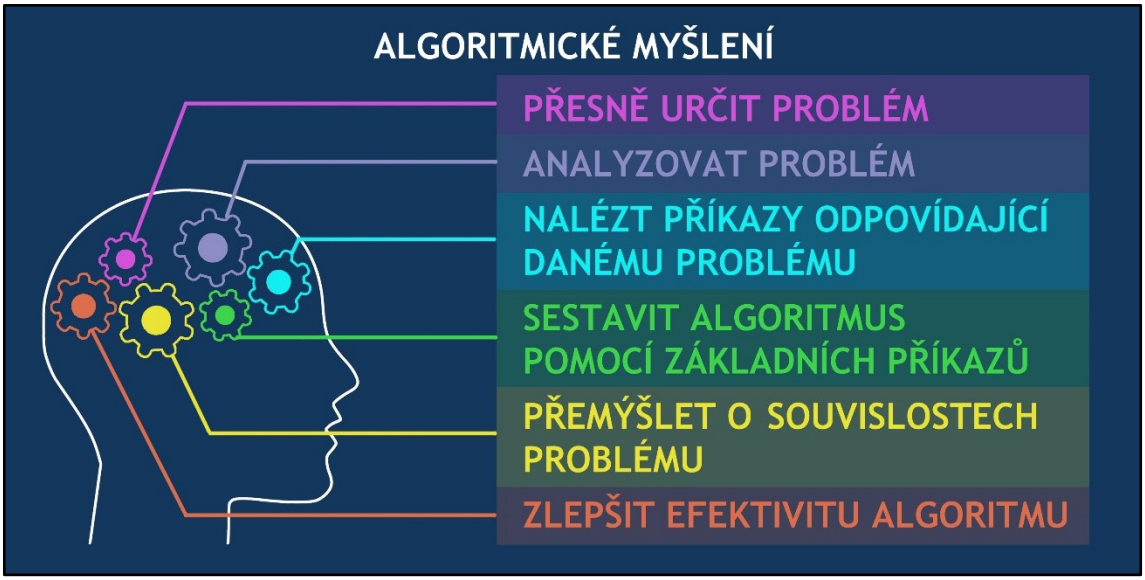
Jednoduché algoritmy lze tvořit ze tří komponent. **Ze sekvencí** (posloupností příkazů), kroků seřazených do konkrétního pořadí, **z podmínek** (větvení), kdy další nejbližší krok závisí na splnění či nesplnění dané podmínky, a **z opakování** (cyklů), kroků či sekvencí kroků, které jsou prováděny s pevným počtem opakování, nebo na základě podmínky, kterou lze umístit na začátek či konec cyklu.

Způsobilost sestavit algoritmus (postup, jak problém řešit), který splňuje všechny tyto podmínky, a dovednost zapsat jej správně pomocí vývojového diagramu či v nějakém programovacím jazyce vyžaduje algoritmické myšlení neboli schopnost algoritmizace.

„Algoritmizace je přístup k vytváření programu. Zabývá se formulací postupů řešení daného problému. Výsledkem algoritmizace je algoritmus, což je posloupnost příkazů popisující řešení daného problému.“ (Dohnal, 2008, str. 10)

V literatuře je možné nalézt celou řadu definicí pojmu algoritmické myšlení. G. Futschek (2006) jej definuje jako soubor schopností, které jsou spojeny s konstruováním algoritmů a porozuměním jim. Uplatní se v nich schopnosti:

- přesně určit problém,
- analyzovat problém,
- nalézt základní příkazy odpovídající danému problému,
- sestavit správný algoritmus pomocí základních příkazů,
- přemýšlet o souvislostech problému,
- zlepšit efektivitu algoritmu. (Futschek, 2006)



Obrázek 3 – Algoritmické myšlení (zdroj autor)

Algoritmické myšlení kromě těchto schopností vyžaduje také další vlastnosti jako abstraktní myšlení, logické myšlení, strukturované myšlení, tvořivost a schopnost řešit problémy. To vede k tomu, že není snadné si dovednost algoritmického myšlení osvojit, a vysvětluje potřebu vhodného didaktického přístupu, zejména u začátečníků. (Futschek, 2006)

V praxi nestačí pouze schopnost algoritmy vytvářet. Je třeba jim opravdu porozumět, abychom je uměli ladit, nacházet v nich chyby či dokázali určit výchozí stav, ke kterému konkrétní algoritmus vede. Teprve pokud jsou ovládány tyto dovednosti, lze vytvářet kvalitní algoritmy a mezi nimi nalézat ty, které jsou:

- *rychlejší (dříve vedou k výsledku),*
- *„levnější“ (kratší, vykonatelné s nižšími náklady),*
- *spolehlivější (ve více případech, častěji vede k výsledku),*
- *bezesporné (nelze je pochopit různými způsoby tak, aby vedli k různým výsledkům).* (Vaníček, 2016, str. 1)

3.3 Postavení algoritmického myšlení v RVP pro základní vzdělávání

Rámcově vzdělávací program pro základní vzdělávání je základní kurikulární dokument ČR vymezující vzdělávací obsah vzdělávacích oblastí. Algoritmické myšlení je v tomto

dokumentu zmíněno ve vzdělávacích oblastech Informační a komunikační technologie a Matematika a její aplikace.

„Vzdělávací oblast Informační a komunikační technologie umožňuje všem žákům dosáhnout základní úrovně informační gramotnosti – získat elementární dovednosti v ovládání výpočetní techniky a moderních informačních technologií, orientovat se ve světě informací, tvořivě pracovat s informacemi a využívat je při dalším vzdělávání i v praktickém životě. Vzhledem k narůstající potřebě osvojení si základních dovedností práce s výpočetní technikou byla vzdělávací oblast Informační a komunikační technologie zařazena jako povinná součást základního vzdělávání na 1. a 2. stupni. Získané dovednosti jsou v informační společnosti nezbytným předpokladem uplatnění na trhu práce i podmínkou k efektivnímu rozvíjení profesní i zájmové činnosti.“ (RVP ZV, 2017, str. 38)

V konkretizaci cílového zaměření vzdělávací oblasti Informační a komunikační technologie zmiňuje RVP ZV algoritmizaci pouze jako jeden z deseti bodů, kterými vzdělávání v oblasti informačních technologií směřuje k utváření a rozvíjení klíčových kompetencí tím, že vede žáka k: *„Schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení.“* (RVP ZV, 2017, str. 38)

Začlenění výuky algoritmizace do hodin informatiky na základních školách je tedy pouze na uvážení jednotlivých škol. Míra rozsahu výuky algoritmického myšlení tak záleží především na doporučení konkrétních učitelů informaticky zaměřených předmětů a je přímo závislá na ředitelích škol, v jejichž kompetenci jsou změny ŠVP, a kteří jsou za jejich obsah přímo odpovědní.

RVP ZV dále zmiňuje algoritmické myšlení v charakteristice vzdělávací oblasti Matematika a její aplikace.

„Vzdělávání klade důraz na důkladné porozumění základním myšlenkovým postupům a pojmům matematiky a jejich vzájemným vztahům. Žáci si postupně osvojují některé pojmy, algoritmy, terminologii, symboliku a způsoby jejich užití. Vzdělávací obsah vzdělávacího oboru Matematika a její aplikace je rozdělen na čtyři tematické okruhy. V tematickém okruhu Čísla a početní operace na prvním stupni, na který na druhém stupni navazuje a dále ho prohlubuje tematický okruh Číslo a proměnná, si žáci osvojují aritmetické operace v jejich třech složkách: dovednost provádět operaci, algoritmické porozumění (proč je

operace prováděna předloženým postupem) a významové porozumění (umět operaci propojit s reálnou situací). Učí se získávat číselné údaje měřením, odhadováním, výpočtem a zaokrouhlováním. Seznamují se s pojmem proměnná a s její rolí při matematizaci reálných situací." (RVP ZV, 2017, str. 30)

V cílovém zaměření vzdělávací oblasti zmiňuje RVP ZV algoritmizaci jako:

- *„rozvíjení paměti žáků prostřednictvím numerických výpočtů a osvojování si nezbytných matematických vzorců a algoritmů"*
- *„vytváření zásoby matematických nástrojů (početních operací, algoritmů, metod řešení úloh) a k efektivnímu využívání osvojeného matematického aparátu"* (RVP ZV, 2017, str. 30)

V rámci doporučeného učiva vymezeného v RVP ZV je zmíněno v bodu:

- *písemné algoritmy početních operací.* (RVP ZV, 2017, str. 32)

V očekávaných výstupech druhého období žák *„řeší jednoduché praktické slovní úlohy a problémy, jejichž řešení je do značné míry nezávislé na obvyklých postupech a algoritmech školské matematiky".* (RVP ZV, 2017, str. 34)

Z uvedeného vyplývá, že RVP ZV (2017) chápe a zmiňuje pojem algoritmizace především jako schopnost osvojit si tradiční matematické postupy. Jedná se tedy spíše o nácvik provádět matematické operace podle předložených historicky ověřených postupů, než hledání vlastních, nových řešení. Obdobně jako v případě informaticky zaměřených předmětů je tak výuka přímo zaměřená na rozvoj algoritmického myšlení pouze na uvážení příslušných škol a učitelů matematiky.

3.4 Výuka algoritmizace a programování

Výuka programování je v České republice, především na základních školách, často opomíjena. Ve výuce informaticky zaměřených předmětů stále přetrvává snaha učit naše žáky především uživatelským dovednostem pro práci s počítačem v prostředí konkrétních programů. Je tomu tak i přesto, že Strategie digitálního vzdělávání do roku 2020 zmiňuje schopnost algoritmického myšlení jako jeden z důležitých prvků informatického myšlení,

jehož rozvoj u žáků je jedním z dílčích cílů tohoto, pro české školství závazného dokumentu. (Fojtík, 2017)

Cílem výuky algoritmického myšlení u žáků není vychovat tisíce mladých programátorů, nýbrž naučit je takovým postupům, které jim umožní uspět v životě v současné společnosti. Efektivní řešení problémů za využití algoritmického myšlení může uspořit lidem nejenom množství času, ale také duševních sil.

Je nesmírně obtížné určit, jakým způsobem lze učit žáky algoritmickému myšlení, jak v nich probouzet tvořivost, kreativitu a konstruktivnost. Jedním z možných způsobů je nechat žáky řešit množství problémů, nejlépe takových, se kterými se běžně setkávají. Podobně, jako v každodenním životě, kdy musí řešit celou řadu problémů algoritmické povahy a nacházet pro ně co nejlepší řešení. Jestliže však máme vytvořit počítačový program, je třeba nalézt řešení co nejvhodnější pro počítač. (Futschek, 2010)

Cíle výuky algoritmizace programování shrnul T. Pitner (2000):

- *„Primárním cílem je naučit algoritmicky myslet, zformulovat zadání problému.*
- *Problém analyzovat nejdříve dekompozicí – rozložením na podproblémy.*
- *Nezbytné je též umět myšlenku dovést k formalizovanému návrhu algoritmu, nejlépe v grafické podobě – vývojové diagramy, struktogramy…, evt. ve formě přesného slovního popisu za použití předem daných „obratů“, tj. vlastně povolených programových struktur.*
- *Přepsání formalizovaného návrhu do podoby programu je technická záležitost, nikoli hlavní cíl výuky.*
- *K samotným algoritmům nedílně patří, ale spíše až „ve druhém pořadí“, datové struktury (objekty).*
- *Vedlejším cílem (na nižších stupních dokonce hlavním cílem) je celkový rozvoj tvořivosti.“* (Pitner, 2000, str. 3)

Podobně, jako je možné rozvíjet a aplikovat matematické dovednosti v rámci dalších předmětů (ve fyzice či chemii) či zdokonalovat písemný projev mimo výuku českého jazyka (v občanské výchově či dějepisu), lze zařadit algoritmizaci i do výuky řady dalších předmětů, například matematiky, fyziky, geografie a dalších. Výuka algoritmického myšlení však zasahuje tak širokou oblast kompetencí, že nelze předpokládat, že ji bude možno v rámci základního vzdělávání do matematiky zařadit tak, aby mohla být účinně vyučována. Cílem výuky algoritmizace na základní škole je především rozvoj obecných dovedností

umožňujících řešit problémy, proto patří tato problematika především do výuky v oblasti ICT. (Vaníček, 2016)

„Algoritmizace je informatická kompetence a nemůže být svěřena do zodpovědnosti jiného vyučovacího předmětu. To neznamená, že v rámci jiných vyučovacích předmětů žáci nemohou s algoritmy pracovat, naopak jejich aplikace je žádoucí. Tíha zodpovědnosti za rozvíjené kompetence však musí ležet na informatice, protože ona se algoritmu komplexně věnuje (stejně jako např. matematika pojmu číslo, i když s čísly se počítá i v jiných oborech) a také protože má prostředky včetně technických, které umožňují algoritmizaci vyučovat.“ (Vaníček, 2016, str. 2)

Především u žáků na primárním a nižším sekundárním stupni vzdělávání je velmi důležité brát zřetel na věk žáků. V tomto období se nejvíce projevují rozdíly u žáků v rozdílných ročnících, ale i úroveň vyspělosti jedinců v rámci jedné třídy. Úroveň vývoje abstraktního myšlení tak může být velmi rozdílná. J. Piaget (1896–1980) ve své teorii kognitivního vývoje definoval čtyři základní vývojová stádia dítěte:

- **senzomotorické stadium (0–2 roky)** – dítě odlišuje sebe od objektů, rozeznává sebe jako aktivního činitele a začíná jednat záměrně,
- **předoperační stadium (2–7 let)** – dítě se učí užívat jazyk, objekty jsou reprezentovány pomocí představ a slov, předměty třídí dle jednoho rysu (červené, hranaté, hebké), myšlení je egocentrické (nevnímá názory druhého),
- **stadium konkrétních operací (7–12 let)** – dítě dokáže logicky přemýšlet v operacích, objektech, událostech; chápe stálost počtu (v 6 letech), množství (v 7 letech) a hmotnosti (v 9 letech); předměty třídí podle různých vlastností a dokáže je logicky seřadit (nejtmavší – nejsvětlejší, největší – nejmenší),
- **stadium formálních operací (12 let a výše)** – dítě dokáže myslet logicky o abstraktních pojmech a systematicky testuje hypotézy; zabývá se abstrakcí, budoucnostní, ideologickými problémy. (Piaget, 2014)

Odlišné vstupní předpoklady žáků, jejich dosavadní zkušenost s prací na počítači i sociální prostředí, ze kterého pocházejí, značně ovlivňují motivaci a přístup žáků k učení. Podstatný vliv na výuku programování má rovněž obsah učiva v ostatních předmětech, především v matematice. Nelze s žáky ve druhém ročníku základní školy řešit problémy, ve kterých je

třeba uspořádat slova podle abecedy, jestliže řazení podle abecedy je součástí výuky až v ročníku třetím. To značně omezuje výběr softwarových nástrojů a úloh, které lze pro výuku programování dětí na prvním stupni základní školy použít. Většina programů je příliš komplikovaná, protože obsahuje příliš mnoho prvků, možností nastavení a rozšiřujících nástrojů. Při programování v profesionálních jazycích je třeba především zvládnout základní syntaxi jazyka, což je pro většinu dětí do věku 12 let nad jejich možnosti. (Lessner, 2015)

G. Futschek se domnívá, že zejména pro začátečníky je vhodné snížit úroveň výuky na takovou, kdy se mohou pojmy algoritmického myšlení naučit přirozeným způsobem.

Proto doporučuje:

- zadávat úkoly, které žáci znají z každodenního života,
- používat pro zápis algoritmů přirozený jazyk,
- použít systém, který umožní testovat algoritmus,
- použít systém, který umožní studentům s algoritmem experimentovat,
- použít systém, který poskytuje okamžitou zpětnou vazbu,
- použít flexibilní systém pro spouštění odlišných algoritmů. (Futschek, 2006)

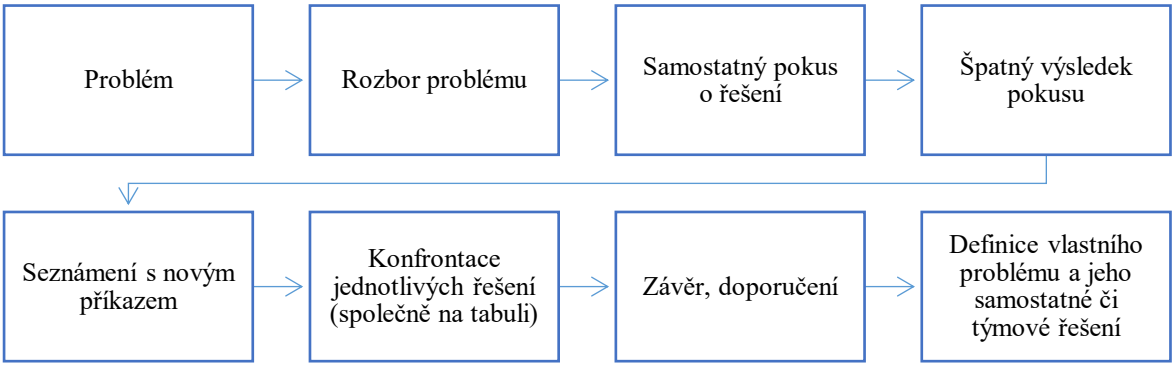
Úroveň chápání konceptů programování žáky ZŠ shrnula E. Kopecká.

Tabulka 4 – Úroveň chápání algoritmických postupů žáky ZŠ (zdroj: Kopecká, 1998 v Pitner, 2000)

4. a 5. ročník	6. a 7. ročník	8. a 9. ročník
Hotové programy	Větvení programu (1 podmínka)	Větší programy
Jednoduché příkazy	Jeden cyklus	Datové struktury
Sekvence příkazů	Proměnná	

„Obecně lze doporučit bez ohledu na objektovost či strukturovanost hned od počátku problémově orientovanou výuku s vysokým podílem samostatné práce. Žáci se s pomocí známých či hotových prostředků snaží problém vyřešit, přicházejí postupně na to, co neumí, učitel jim to ve vhodnou chvíli odtajní a objasní a oni to vzápětí použijí v praxi. Výhodné je zpočátku vždy detailně rozebrat určité řešení včetně rozboru a napsání kódu (algoritmu) přesně na tabuli.“ (Pitner, 2000, str. 4)

Postup problémově orientované výuky může být zhruba takový:



Obrázek 4 – Postup problémově orientované výuky (dle Pitner, 2000)

R. Pecinovský varuje před výukou pomocí „AHA“ příkladů. Domnívá se, že práce s ukázkami konkrétních problémů přibližují funkci probíraných kódů, ale učícím se mohou unikat souvislosti s dalšími částmi programu. Ti pak v praxi nejsou schopni začlenit probíraný koncept do kontextu celého programu. R. Pecinovský upřednostňuje opakovanou práci s menšími, jednoduchými programy, které žáci dobře pochopí a jsou schopni je průběžně modifikovat, aby bylo dosaženo co nejlepších výsledků. To jim umožní pracovat nejen s ukázkami a částmi kódů, ale celými programy. Dovolí jim soustředit se na aktuální problém a nezabývat se již probraným kódem okolo. (Pecinovský, 2011)

Jako nejvhodnější věk žáků pro započetí výuky programování považuje R. Pecinovský 10–12 let, což odpovídá páté až šesté třídě základní školy. Domnívá se, že začít s výukou až v osmé či deváté třídě, je příliš pozdě.

I. Kalaš (2017) varuje před předčasnou výukou programování v konkrétním jazyce. Domnívá se, že: „*Primární programování, tedy programování pro žáky ve věku od 5 do 11 let, poskytuje natolik bohatou paletu vývoji přiměřených námětů a informatických konstruktů, že začít používat konkrétní jazyk, například Scratch, dříve, než je vhodné, představuje značné riziko. Pokud začneme s jazykem a prostředím příliš brzy, vytlačíme tím z vymezeného času příležitost pro rozvoj základních poznatků a aktivit, které tvoří vývojově přiměřený základ a obsah primárního programování. Navíc ohrozíme zájem žáků o práci v tomto prostředí v okamžiku, kdy již dosáhnou vhodného věku. Pak mohou mít pocit, že… tohle jsme již přece dělali.*“ (Kalaš, 2017, str. 23)

Současně ale I. Kalaš varuje i před opačným rizikem: začít s programováním v konkrétním jazyce příliš pozdě. Pak může dojít k situaci, kdy žáci budou považovat příslušné prostředí a nástroje za příliš „dětské“. Pokud sami žáci budou považovat pracovní prostředí za vhodné pro mladší žáky, může je to od samotného programování odrazovat. (Kalaš, 2017)

V posledních letech proběhla řada projektů, které se zabývaly výzkumy v oblasti výuky základů programování. V tabulce č. 5 jsou vybrané příspěvky z konferencí Didinfo 2015–2017, které shrnují zkušenosti autorů s využitím vývojového prostředí Scratch. Věk žáků, kteří se projektů účastnili, se pohyboval v rozmezí od 8 do 16 let.

Tabulka 5 – Vybrané projekty zmiňující výuku programování v Scratch

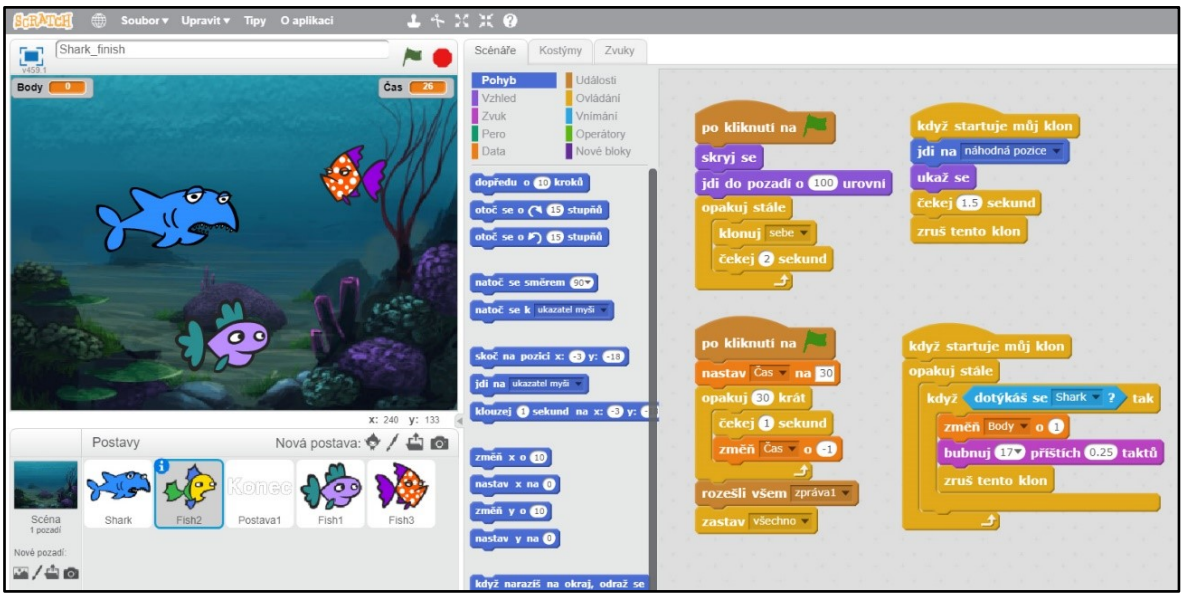
Název příspěvku	Autoři	Rok	Věk žáků	Zdroj
Príprava budúcich učiteľov informatiky a nové trendy vo výučbe programovania	J. Majherová H. Palásthy V. Králík P. Lajčiak	2014	1. a 2. st. ZŠ	Majherová, 2014
Skúsenosti s programovaním na základnej škole	Z. Hájek P. Vasaráb	2015	14-15	Hájek, 2015
Programování ve Scratch badatelsky orientovaným přístupem	J. Vaníček	2015	14-15	Vaníček, 2015
První ohlédnutí za výukou základů programování ve Scratch na ZŠ aneb Čím nás žáci překvapili i zaskočili, co musíme příště dělat jinak	M. Černochová H. Šandová	2015	11-12	Černochová, 2015
Výuka programování pomocí robota Ozobot	R. Fojtík	2017	8-10	Fojtík, 2017
Rozvíjanie čitateľskej gramotnosti prostredníctvom detského programovacieho jazyka Scratch	M. Nagy	2017	9-10	Nagy, 2017
ScratchMaths: vzdelávací obsah a princípy tvorby	I. Kalaš	2017	9-11	Kalaš, 2017
Interaktívne učebné materiály vo vyučovaní algoritmizácie	J. Majherová H. Petrušková P. Valuška	2017	2. st. ZŠ	Majherová, 2017

Otázku, jaké zvolit metody výuky začátečníků základům programování si musí klást každý učitel informatiky. Zásadní roli zde, obdobně jako v dalších předmětech, hraje motivace. Jednou z možností jak žáky při výuce motivovat je vyučování prostřednictvím programování jednoduchých her. Ve prospěch této metody hovoří pozitivní praktické zkušenosti některých učitelů. Studenti rychleji pochopí základní objektové pojmy a navíc jsou víc motivováni k tvořivé činnosti. (Kanáliková, 2015)

Jedním z nástrojů, který umožňuje tvorbu jednoduchých interaktivních příběhů, her a animací i začátečníkům, a současně je vhodný pro výuku základů programování, protože podněcuje tvořivé myšlení, je Scratch.

3.5 Scratch

Scratch je vizuální programovací jazyk. Jedná se o volně šiřitelný software, který byl vyvinut v letech 2003 až 2007 na Massachusetts Institute of Technology týmem pod vedením M. Resnicka. Díky jednoduchosti a intuitivnímu prostředí Scratch dovoluje programovat i žákům na prvním stupni základních škol. Program se sestavuje z připravených bloků, které do sebe zapadají a vytváří tak posloupnost kódu. Jedná se o nástroj, který umožňuje začátečníkům soustředit se na experimentování s částmi kódu a samotnou tvorbu programů, protože není nutné, aby uměli kódovat v textovém jazyce. Pomocí Scratch mohou žáci vytvářet vlastní interaktivní animace, příběhy, prezentace či hry. Od roku 2013 lze díky cloudovému řešení Scratch používat bez instalace přímo v prohlížeči, tvořit a sdílet vytvořené programy přímo na stránkách https://Scratch.mit.edu/.



Obrázek 5 – Prostředí programu Scratch

Program je plně lokalizován do českého jazyka. Přestože se jedná o software na principu přemisťování virtuálních objektů táhni a pusť (drag&drop), které reprezentují konkrétní kódové příkazy, lze v něm vytvářet i velmi sofistikované programy. Scratch lze využívat

rovněž v off-line verzi, kterou lze instalovat do počítače, což umožňuje vytvářet programy bez přístupu k internetu. Je volně dostupný a lze jej používat pod operačními systémy Windows, Mac OS X i Linux. Pro analýzu programů Scratch slouží nástroj Dr. Scratch, který provádí jejich automatickou analýzu.

Scratch není pouze programovacím jazykem, ale také rozsáhlou mezinárodní on-line komunitou. Děti i dospělí z celého světa mohou nejen programovat, ale v případě aktivního uživatelského účtu také sdílet své příběhy, animace či hry. Přihlášení uživatelé přispívají nejenom sdílením svých projektů, ale také jejich komentováním, čímž poskytují tvůrcům zpětnou vazbu. Sdílené projekty jsou volně dostupné a editovatelné, což je umožňuje modifikovat a na již hotových programech se tak rychleji učit.

Pro využití Scratch ve školním vzdělávání vznikla celosvětová komunita učitelů sdružená kolem portálu ScratchEd. Scratch je používán v rámci formálního, ale i neformálního učení. Na portálu Scratch lze sdílet projekty, hledat zdroje, diskutovat a nacházet osoby se společnými zájmy a zkušenostmi.

Jak řekl M. Resnick *„Schopnost kódovat počítačové programy je důležitou částí gramotnosti v dnešní společnosti. Když se lidé učí programovat ve Scratch, naučí se důležité strategie pro řešení problémů, navrhování projektů a plánování komunikace.“* (Lifelong Kindergarten Group, MIT Media Lab., 2018)

Scratch je proto ideálním edukačním programovacím prostředím pro rozvoj kompetencí k řešení problémů a informatického, tedy rovněž algoritmického myšlení. Pro tvorbu programů využívá Scratch příkazů, rozdělených do kategorií Pohyb, Vzhled, Pero, Zvuk, Data, Události, Ovládání, Vnímání, Operátory a Nové bloky. Programovací koncepty využívané v Scratch, jsou uvedeny v tabulce č. 6 a korespondují s běžnými programovacími koncepty. Jedná se o sekvence, cykly, podmínky, proměnné, paralelizaci, události, zprávy, náhodná čísla, Booleovu logiku, interakce v reálném čase a návrhy uživatelského rozhraní.

Tabulka 6 – Koncepty ve Scratch (převzato a upraveno z Programming Concepts and Skills Supported in Scratch, Lifelong Kindergarten Group, MIT Media Lab., 2017)

Koncept	Popis	Příklad
Sekvence	Při tvorbě programu v aplikaci Scratch je nutné systematicky uvažovat o pořadí kroků.	
Cyklus	Příkazy Opakuj stále a Opakuj lze použít pro opakování bloku příkazů.	
Podmínky	Pro kontrolu podmínek slouží příkazy KDYŽ a KDYŽ – JINAK.	
Proměnná	Kategorie proměnná umožňuje vytvořit novou proměnnou a použít ji v programu.	
Paralelizace (vlákna)	Spuštění dvou různých bloků příkazů ve stejném čase umožňuje paralelní, synchronizovaný běh více vláken programu.	

Události	Kliknutí na objekt, stisknutí klávesy, obdržení zprávy jsou příklady událostí, které iniciují další běh kódu jako reakci na tuto událost.	
Zprávy	Umožňují vytvářet vlastní události, rozesílat je všem postavám a pomocí události „Po obdržení zprávy“ na ně reagovat.	
Náhodná čísla	Umožňují generování náhodného čísla daného rozsahem hodnot.	
Booleova logika	Operátory A, NEBO, NENÍ patří do Booleovy logiky, která se zabývá logickými operacemi na množině {0;1}.	
Interakce v reálném čase	Souřadnice myši X a Y nebo hlasitost mohou být použity jako dynamický vstup pro interakci v reálném čase.	
Návrh uživatelského rozhraní	Kliknutím na konkrétní objekt lze vytvořit vlastní tlačítko.	

4 PRAKTICKÁ ČÁST

Cílem praktické části je zjistit, do jaké míry aktivity se Scratch přispívají k rozvoji algoritmického myšlení žáků ZŠ. Pro tento účel bylo zapotřebí:

1. navrhnout dvě sady testových úloh pro ověření dopadu aktivit Scratch na utváření dovedností žáků algoritmicky myslet;
2. uspořádat a vyhodnotit pedagogický experiment, jehož základem je výuka Scratch zaměřená na rozvoj algoritmického myšlení žáků 5. a 6. ročníku ZŠ.

4.1 Charakteristika vzorku žáků

Pedagogický experiment probíhal na jedné běžné základní škole v centru Prahy. Jedná se o středně velkou školu, která má v současnosti 250 žáků, s rozšířenou výukou tělesné výchovy, informatiky a ekologie. Vzhledem k poloze školy je značné procento žáků z řad cizinců, především z Vietnamu, Číny, Ruska a Ukrajiny. Znalost českého jazyka u těchto žáků je velice rozdílná a schopnost porozumět obsahu výuky je u některých z nich velmi omezená. To se projevuje především v odborných předmětech, a tedy zásadním způsobem ovlivňuje i výuku informaticky zaměřených předmětů. Z tohoto důvodu nebyly výsledky integrovaných žáků, u nichž byla v době experimentu snížena schopnost porozumět výkladu, zařazeny do závěrečného hodnocení experimentu.

Pedagogický experiment proběhl od 4. října 2017 do 20. dubna 2018 jako součást výuky povinného předmětu Informačně technologický základ, který je na škole vyučován v rámci vzdělávací oblasti Informační a komunikační technologie. Příslušná hodinová dotace této vzdělávací oblasti činí jednu hodinu týdně na prvním stupni a jednu hodinu týdně na stupni druhém. Předmět je vyučován v 5. a 6. ročníku. Vzhledem k počtu žáků ve třídách a omezené kapacitě počítačové pracovny (18 pracovních stanic) jsou žáci tříd, které přesahují tento počet, na výuku tohoto předmětu pro celý školní rok rozdělováni do více skupin. Ve školním roce 2017/2018 byli žáci 5. třídy rozděleni do dvou skupin a žáci 6. třídy rovněž. Vznikly tak 4 pracovní skupiny účastnící se experimentu a pro účely této práce byly označeny písmeny A, B, C a D (viz Tabulka 7).

Tabulka 7 – Rozdělení skupin

Třída	Skupina	Počet žáků	Označení
5. třída	Skupina 1	11	A
	Skupina 2	12	B
6. třída	Skupina 3	13	C
	Skupina 4	12	D

Protože ŠVP školy nezahrnoval výuku základů programování, byla nutná jeho úprava. Nový školní vzdělávací plán byl řádně schválen radou školy.

S ohledem na cíle práce byl vytvořen dotazník (DOT 01, viz Příloha A), s jehož pomocí se uskutečnilo šetření, které mělo za úkol zmapovat a popsat charakteristiku jednotlivých skupin žáků zařazených do pedagogického experimentu. Pomocí dotazníku se zjišťovala doba, kterou žáci stráví u počítače, způsoby využití počítačů a vztah žáků k programování. Přípravě dotazníku předcházel průzkum činností, které žáci vykonávají na počítačích. Získané odpovědi lze rozřadit do pěti kategorií: hraní počítačových her, komunikace pomocí sociálních sítí, příprava do školy, zábava mimo hry (hudba, videa, brouzdání po internetu) a tvořivá činnost (kreslení, psaní, editace fotografií, programování apod.). S ohledem na cíle práce byla kromě těchto pěti kategorií do dotazníku rovněž zařazena otázka zaměřená přímo na dobu věnovanou programování. V konečné verzi dotazník obsahoval celkem 24 otázek, na jejichž zodpovězení měli žáci jednu vyučovací hodinu v rozsahu 45 minut. V rámci těchto hlavních témat byly žákům položeny otázky zkoumající:

- dobu strávenou u počítače
  - počet dnů v týdnu, kdy žáci využívají počítač
  - počet hodin ve dnech, kdy na počítači žáci pracují
- druh vykonávaných činností žáků na počítači
  - hry
  - komunikace, sociální sítě
  - příprava do školy
  - zábava mimo hry (hudba, videa, brouzdání po internetu)
  - tvořivé činnosti (kreslení, psaní, editace fotografií, programování apod.)

- hodnocení vlastních znalostí a dovedností v oblasti ICT
- rodinné prostředí žáků
    - kontrola času a způsobu používání počítače
    - přístup žáků k technologiím
- vztah žáků k programování
    - porozumění pojmům souvisejícím s programováním
    - zkušenosti s programováním.

Dotazníkového šetření se zúčastnilo celkem 48 žáků z 5. a 6. ročníku ve věku 10–13 let. Dotazník byl zadán v tištěné podobě v týdnu od 25. do 29. září 2017. Aby nedocházelo ke zkreslování odpovědí žáků díky nejasnostem, byly žákům před jeho vyplněním i v jeho průběhu objasněny termíny v něm obsažené.

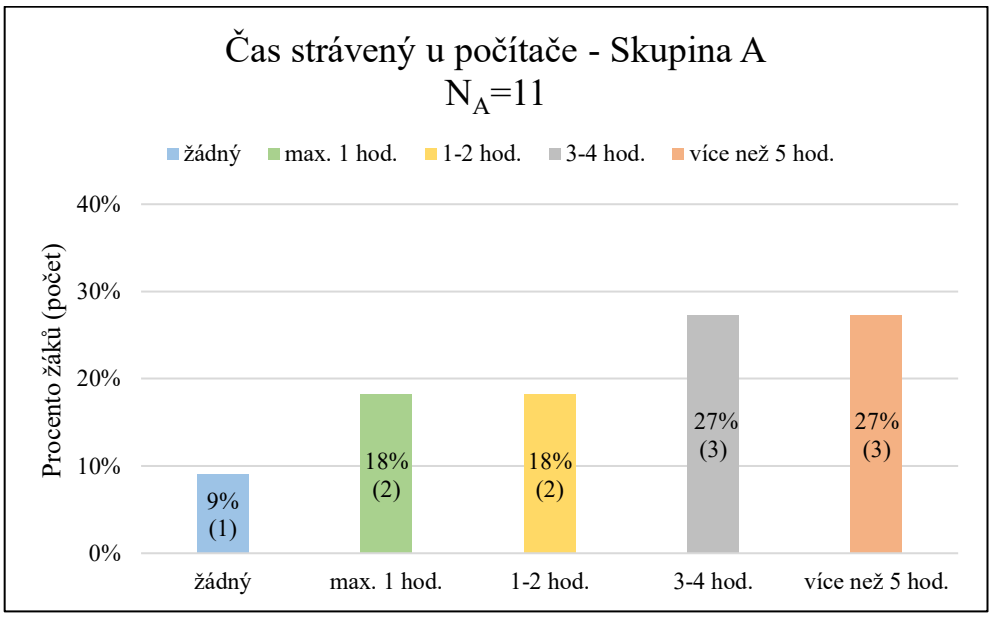
Z důvodu zachování anonymity žáků byly dotazníky vyplňovány pod číselným kódem. Současně však byla zachována možnost vstupní dotazníky spárovat se výstupními dotazníky. Obdobně bylo postupováno i v případě testových úloh a pracovních listů.

Data získaná dotazníkovým šetřením byla vyhodnocena běžnými statistickými metodami a zásadní informace, které z dotazníku vyplynuly, byly znázorněny pomocí grafů.

4.1.1 Charakteristika žáků – skupina A

Skupina A je složena z žáků 5. třídy ve věku 10 a 11 let. Počet žáků je 11 z toho 8 chlapců a 3 dívky. Jeden žák této skupiny uvedl, že mimo školu nemá možnost využívat počítač a jeden žák jej využívá pouze jeden den v týdnu. Ostatní žáci používají počítač nejméně čtyři dny v týdnu a téměř polovina žáků této skupiny (46 % žáků) používá počítač každý den.

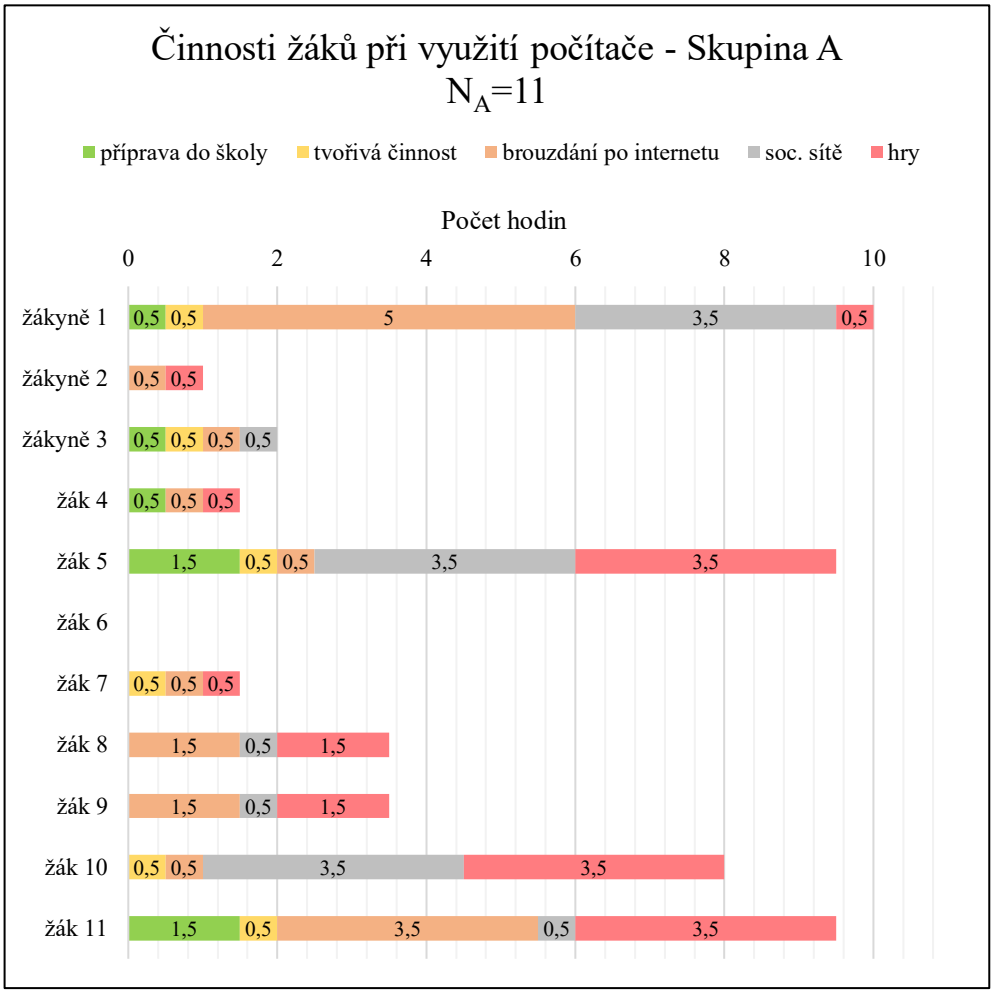
Průměrně tedy žáci této skupiny používají počítač 5 dní v týdnu. Více než 55 % žáků skupiny A uvedlo, že používá počítač (tablet, notebook) více než 3 hodiny denně, z toho polovina více než 5 hodin. Průměrně denně žáci mimo školu tráví u počítače 1 h 52 min. Své IT znalosti hodnotí žáci průměrně, a stejně je tomu i u IT dovedností.



Graf 1 – Čas strávený u počítače – Skupina A

Graf 2 ukazuje, že s výjimkou žáka č. 6, který výpočetní techniku nepoužívá, a žákyně č. 3, která rozděluje mezi vykonávané činnosti stejné množství času, u všech zbývajících členů skupiny velmi výrazně převyšuje čas věnovaný zábavě oproti tvůrčím činnostem a přípravě do školy.

Jako tvůrčí činnost při práci na počítači žáci skupiny A nejčastěji uvedli kreslení, tvůrčí psaní a editaci fotografií. Jako další činnost jeden z žáků uvedl nahrávání a editaci videí.



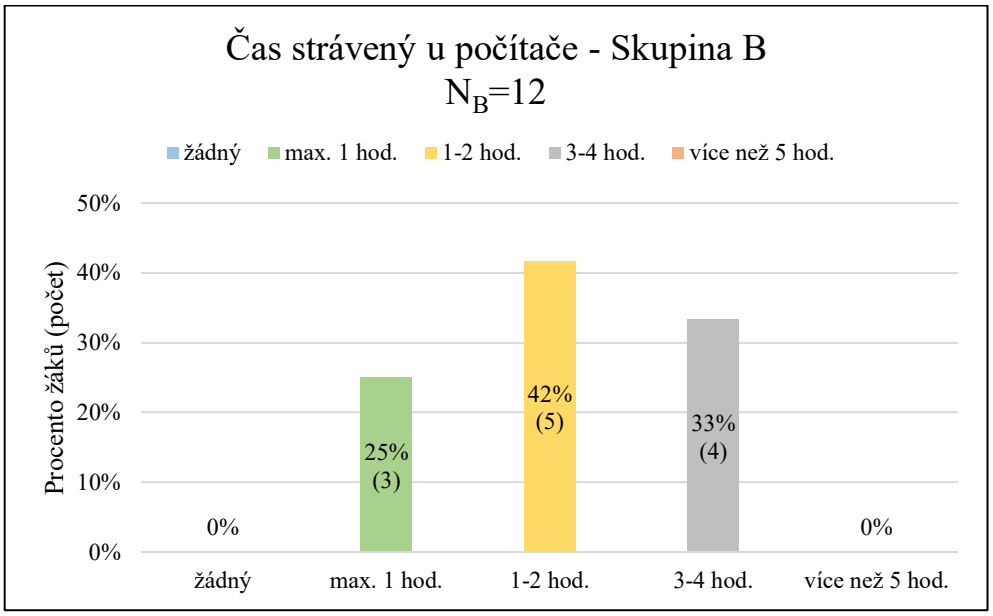
Graf 2 – Činnosti žáků při využití počítače – skupina A

Většina rodičů žáků skupiny A omezuje čas dětí u počítače; sedm žáků (64 %) skupiny A uvedlo, že jejich rodiče kontrolují, kolik času stráví u počítače, a pouze tři žáci (27 %) uvedli, že je rodiče nekontrolují. Pouze čtyři žáci (36 %) uvedli, že jejich rodiče kontrolují druh činnosti, kterou na počítači vykonávají.

Sedm žáků (64 %) skupiny A pochází z rodin, kde oba z rodičů využívají ve svém povolání počítač, ve čtyřech případech (36 %) z rodin, kdy ani jeden z rodičů ve svém povolání počítač nevyužívá. Pouze jeden žák uvedl, že vlastní programovatelnou hračku. Čtyři žáci (36 %) skupiny A již v minulosti programovali v aplikaci Game Maker Studio 2.

4.1.2 Charakteristika žáků – skupina B

Skupina B je složena z žáků 5. třídy ve věku 10 a 11 let. Počet žáků je 12, z toho 10 chlapců a 2 dívky. Všichni žáci skupiny B používají mimo školu počítač. Jeden žák používá počítač pouze dva dny v týdnu, ostatní nejméně čtyřikrát v týdnu. Polovina žáků používá počítač denně. Průměrně tedy žáci používají počítač 6 dní v týdnu. Ve skupině B nejsou žáci, kteří by počítač nevyužívali vůbec, ani žáci, kteří by práci na počítači věnovali více než 4 hodiny denně (viz Graf 3). Průměrně denně žáci tráví mimo školu u počítače 1 h 52 min. Své IT znalosti hodnotí žáci průměrně, a stejně je tomu i u IT dovedností.



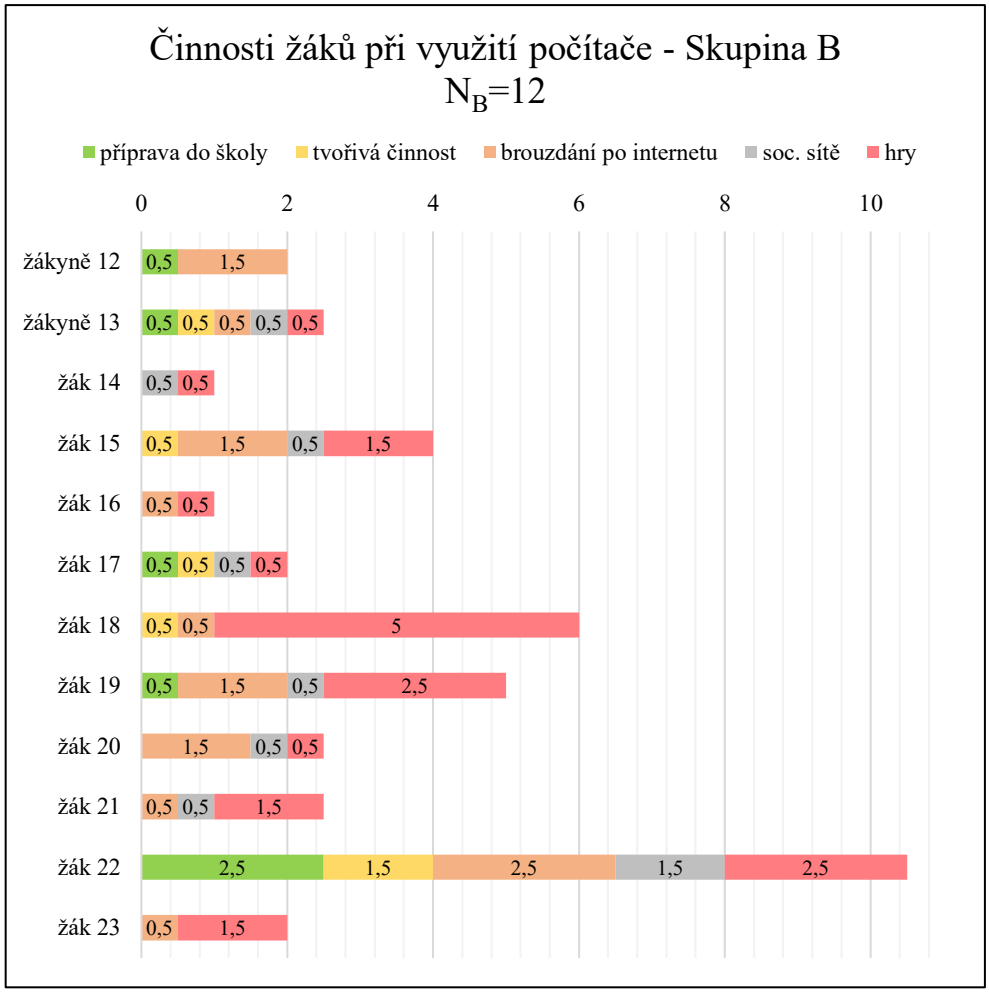
Graf 3 – Čas strávený u počítače – Skupina B

Graf 4 ukazuje, že ve skupině B je rozložení druhu činnosti obdobné, jako v případě žáků skupiny A. Výjimkou je žák č. 17, který rozděluje mezi jednotlivé činnosti stejné množství času. U všech zbývajících členů skupiny podobně jako v případě skupiny A převyšuje čas věnovaný zábavě oproti času vymezenému tvořivým činnostem a přípravě do školy.

Jako tvořivou činnost při práci na počítači žáci skupiny B nejčastěji uvedli kreslení, na druhém místě natáčení a editaci videí. Dále pak editaci fotografií a tvůrčí psaní.

Většina rodičů žáků skupiny B omezuje čas svých dětí u počítače; osm žáků (67 %) skupiny B uvedlo, že jejich rodiče kontrolují množství času, které stráví u počítače a pouze čtyři žáci (33 %), že nikoli. Oproti tomu činnost, kterou žáci vykonávají na počítači, již tolik

pod kontrolou není. Pouze čtyři žáci (33 %) skupiny B uvedli, že jejich rodiče kontrolují druh činnosti, kterou na počítači vykonávají.



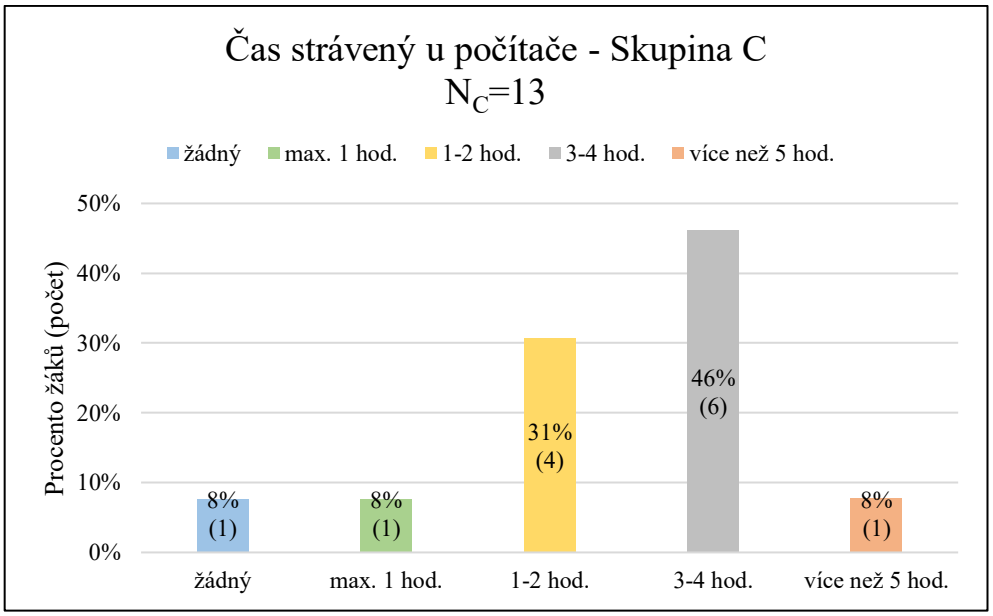
Graf 4 – Činnosti žáků při využití počítače – skupina B

Čtyři žáci (33 %) skupiny B pochází z rodin, kde oba z rodičů využívají ve svém povolání počítač, pět (42 %) z rodin, kde ve svém povolání počítač využívá jeden z rodičů. V zaměstnání nevyužívají počítač rodiče tří (25 %) žáků. Dva z rodičů žáků se zabývají programováním. Pouze jeden žák uvedl, že vlastní programovatelnou hračku a v minulosti programoval v aplikaci Lego Mindstorms.

4.1.3 Charakteristika žáků – skupina C

Skupina C je složena z žáků 6. třídy ve věku 11 a 12 let. Počet žáků je 13, z toho 8 chlapců a 5 dívek. Jedna žákyně této skupiny uvedla, že mimo školu nemá možnost využívat počítač. Všichni ostatní členové skupiny používají počítač minimálně čtyři dny v týdnu, deset žáků (77 %) minimálně šest dní v týdnu. Devět (61 %) žáků této skupiny používá počítač denně. Průměrně tedy žáci používají počítač 6 dní v týdnu.

Celkem 7 žáků (54 %) skupiny C uvedlo, že používá počítač (tablet, notebook) více než 3 hodiny denně, z toho jeden více než 5 hodin. Průměrně denně žáci mimo školu tráví u počítače 2 h 6 min. Své IT znalosti hodnotí žáci průměrně, a stejně je tomu i u IT dovedností.

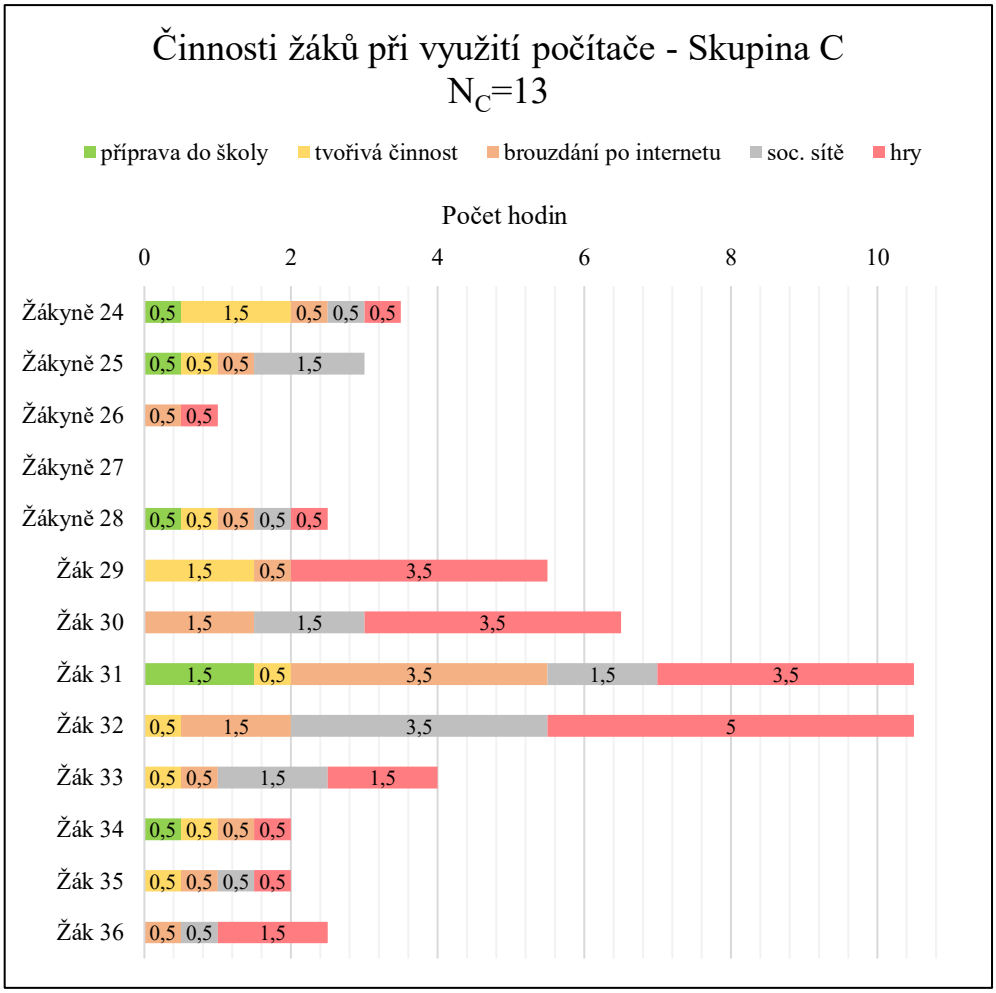


Graf 5 – Čas strávený u počítače – Skupina C

Graf 6 ukazuje, že s výjimkou žákyně č. 27, která výpočetní techniku nepoužívá, u všech zbývajících členů skupiny velmi výrazně převyšuje čas věnovaný zábavě, oproti tvořivým činnostem a přípravě do školy.

Příprava do školy tvoří u žáků skupiny C pouhých 7 % a tvořivá činnost 12 % z celkového času stráveného u počítače. Jako tvořivou činnost při práci na počítači žáci skupiny C nejčastěji uvedli editaci fotografií a kreslení. Jako další činnost jeden z žáků uvedl nahrávání

a editaci videí a jeden žák z této skupiny má z minulosti zkušenosti s programováním v programu Game Maker.



Graf 6 – Činnosti žáků při využití počítače – Skupina C

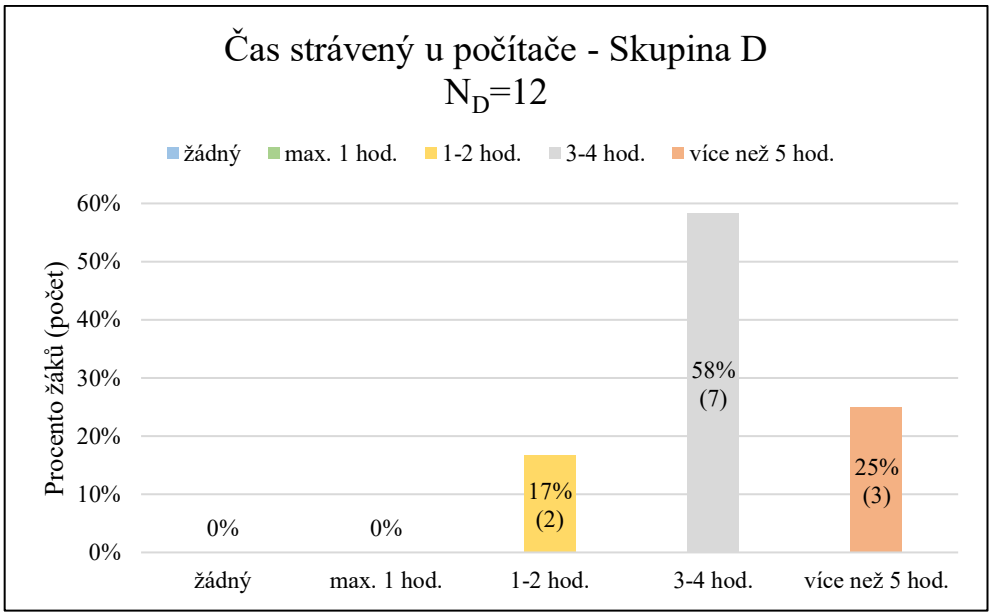
Většina rodičů žáků skupiny C omezuje čas dětí u počítače; devět žáků (69 %) skupiny uvedlo, že jejich rodiče kontrolují množství času, který stráví u počítače, a pouze tři žáci (23 %), že nikoli. Oproti tomu činnost, kterou žáci vykonávají na počítači, již tolik pod kontrolou není. Pouze čtyři žáci uvedli, že jejich rodiče kontrolují druh činnosti, kterou na počítači vykonávají.

Pět žáků (38 %) pochází z rodin, kde oba z rodičů využívají ve svém povolání počítač, v osmi případech (62 %) z rodin, kdy ani jeden z rodičů ve svém povolání počítač nevyužívá. Pouze jeden žák uvedl, že vlastní programovatelnou hračku. Jeden rodič žáka z této skupinu pracuje jako programátor.

4.1.4 Charakteristika žáků – skupina D

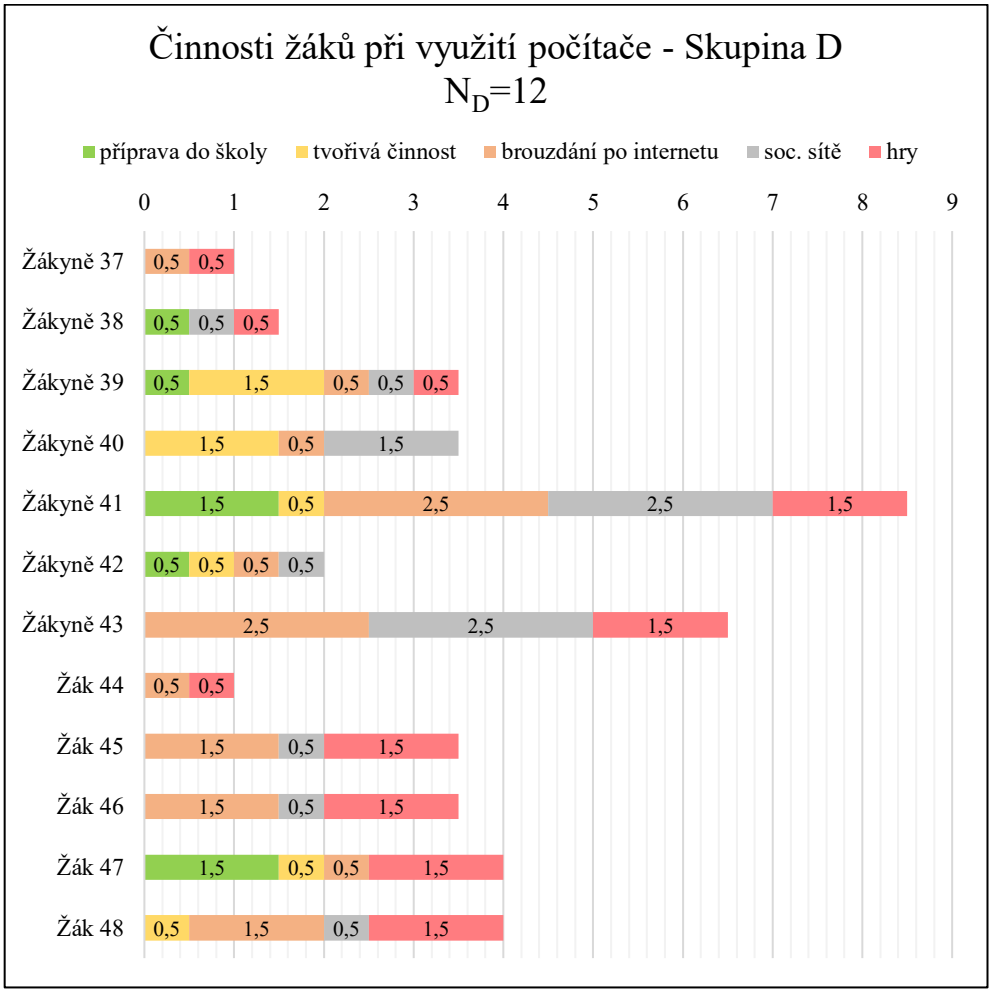
Skupina D je složena z žáků 6. třídy ve věku 11 a 12 let. Počet žáků je 12, z toho 5 chlapců a 7 dívek. Jeden žák skupiny D používá počítač pět dnů v týdnu, pět žáků (42 %) šest dnů v týdnu a šest žáků (50 %) denně. Průměrně tedy žáci používají počítač 6 dní v týdnu.

Všichni žáci skupiny uvedli, že používají počítač (tablet, notebook) minimálně hodinu denně. Dva žáci (17 %) jednu až dvě hodiny denně, sedm žáků (58 %) tři až čtyři hodiny denně a tři žáci (25 %) více než 5 hodin. Průměrně denně žáci mimo školu tráví u počítače 3 h 30 min. Skupina D se tedy od ostatních výrazně odlišuje, neboť její žáci věnují o téměř 1 h a 30 minut denně více času práci na počítači, než členové ostatních skupin. Své IT znalosti hodnotí žáci průměrně, a stejně je tomu i u dovedností.



Graf 7 – Čas strávený u počítače – Skupina D

Graf 8 ukazuje, že u čtyř žáků (33 %) skupiny D je doba věnovaná přípravě do školy a tvořivým činnostem vyrovnaná době věnované oddechovým a zábavným činnostem. U ostatních osmi žáků (67 %) skupiny výrazně převyšuje čas věnovaný zábavě, oproti tvořivým činnostem a přípravě do školy. Pět žáků (42 %) skupiny D nevěnuje přípravě do školy a tvořivým činnostem na počítači žádný čas.



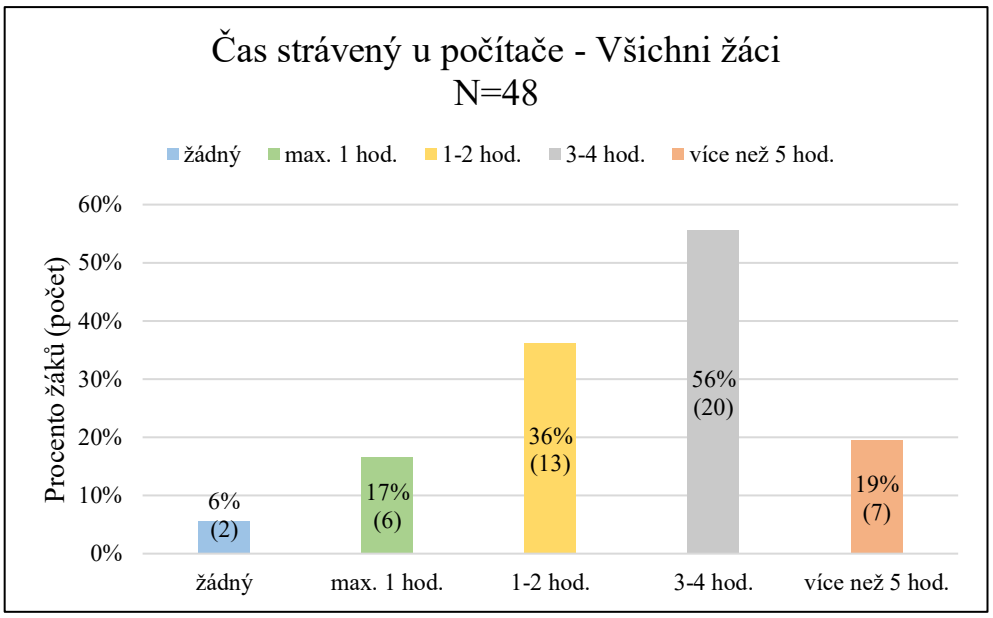
Graf 8 – Činnosti žáků při využití počítače – Skupina D

Příprava do školy tvoří u žáků skupiny D 13 % a tvořivá činnost 14 % z celkového času stráveného u počítače. Jako tvořivou činnost při práci na počítači žáci skupiny D nejčastěji uvedli editaci fotografií či videa a kreslení.

Většina rodičů žáků skupiny D omezuje čas dětí u počítače; osm žáků (67 %) skupiny uvedlo, že jejich rodiče kontrolují množství času, který stráví u počítače a pouze čtyři (33 %), že nikoli. Oproti tomu činnostem, které žáci vykonávají na počítači, již rodiče většinou pozornost nevěnují. Pouze dva žáci (17 %) uvedli, že jejich rodiče kontrolují druh činnosti, kterou na počítači vykonávají.

4.1.5 Charakteristika celého vzorku žáků

Ze získaných údajů vyplývá, že naprostá většina žáků využívá v domácím prostředí počítač. Z celkového počtu 48 žáků pouze dva (4 %) uvedli, že nemají možnost mimo školu pracovat na počítači. Průměrný počet dnů v týdnu, kdy žáci mimo školu používají počítač, je šest. Přestože dvě třetiny žáků v dotazníku uvedly, že jejich rodiče omezují čas, kdy smí používat počítač, množství času, které žáci počítačům věnují, je poměrně značné.

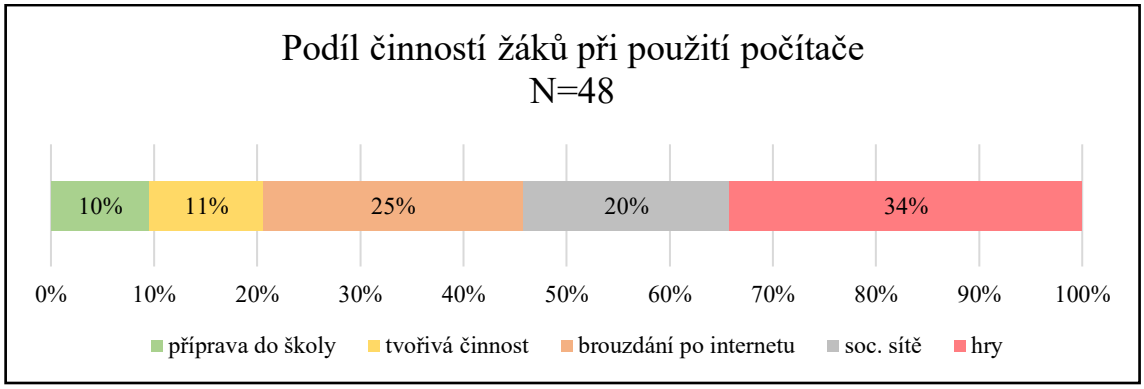


Graf 9 – Čas strávený u počítače – všichni žáci

Maximálně jednu hodinu denně používá počítač pouze 17 % žáků a jednu až dvě hodiny denně 36 % žáků. Celkem 56 % žáků uvedlo, že stráví denně u počítače tři až čtyři hodiny denně a 19 % žáků uvedlo čas více než pět hodin denně. V průměru žáci všech skupin tráví u počítače mimo školu 2 h 20 min.

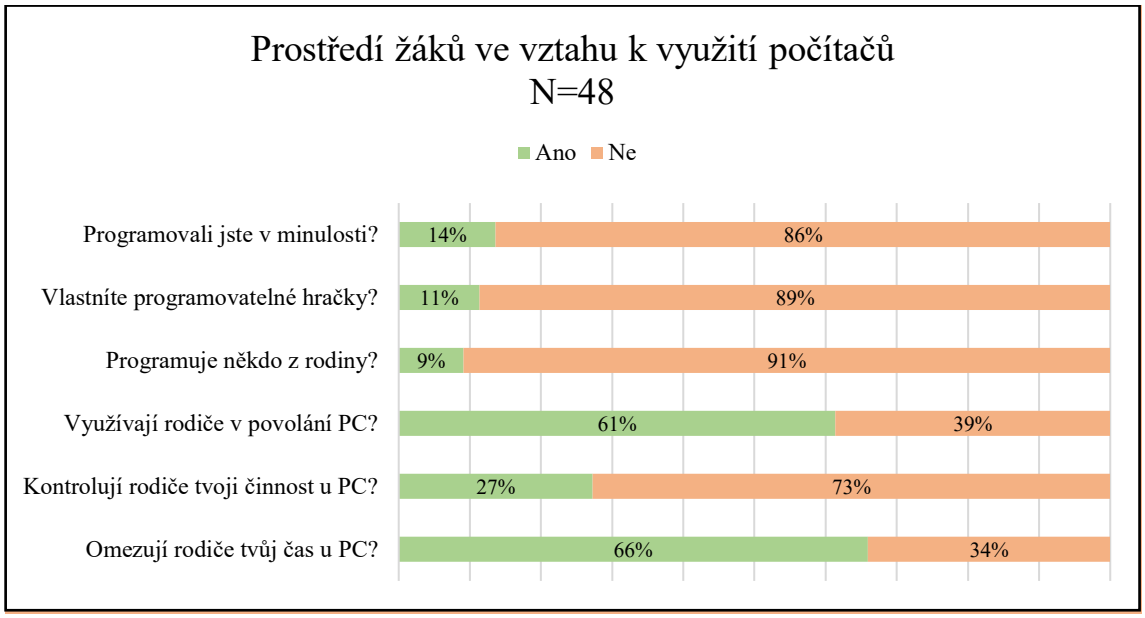
Jak ukazuje Graf 11, přestože většina rodičů kontroluje čas žáků, který věnují počítači, pouze 27 % žáků uvedlo, že kontrolují i druh činnosti, kterou žáci na počítači vykonávají. Celkem 34 % času stráveného u počítače věnují žáci zařazení do experimentu hraní her. Dále následuje brouzdání po internetu (sledování videa, poslouchání hudby) a komunikace pomocí sociálních sítí. Pouhých 11 % času věnují tvořivým činnostem, mezi kterými

převažují především kreslení a editace fotografií či videí. Nejméně využívají žáci počítač pro přípravu do školy – pouhých 10 % času.



Graf 10 – Podíl činností žáků při použití počítače

Žáci pocházejí z prostředí, kde ve většině případů alespoň jeden z rodičů používá ve svém povolání počítač. Čtyři žáci (8,3 %) uvedli, že jeden z rodičů je zaměstnán jako programátor. Pět žáků (10 %) vlastní programovatelné hračky, ve čtyřech případech Lego Mindstorm, v jednom případě se jedná o blíže nespecifikovaného robota. Pět žáků (10 %) rovněž uvedlo, že v minulosti již programovali v aplikaci Game Maker. S programem Scratch neměli žáci před započetím výuky žádné zkušenosti.



Graf 11 – Prostředí žáků ve vztahu k využití počítačů

4.2 Návrh úloh pro zjištění úrovně algoritmického myšlení

Testové úlohy jsou určeny pro žáky 5. a 6. tříd základní školy a odpovídající ročníky víceletých gymnázií. Účelem testu je zmapování úrovně schopnosti žáků používat algoritmické myšlení. Pro tento účel byly vytvořeny úlohy zaměřené na algoritmizaci, logiku, a programování (viz Příloha D). Jsou řazeny do tří kategorií obtížnosti. Obsahují zadání, odpovědi, vysvětlení správné odpovědi, důvod, proč je lze řadit mezi úlohy algoritmické a případnou strategii jejich řešení. Inspirací pro tyto úlohy se staly algoritmicky zaměřené úlohy soutěže iBobr na https://www.ibobr.cz/ (Bobřík informatiky), která ověřuje schopnost informatického myšlení žáků.

4.2.1 Pilotní ověření

Pro ověření funkčnosti testů a odhalení případných chyb byly úlohy pilotně ověřeny ve dvou fázích. Nejprve testy vyzkoušely děti ve věku 10–13 let na letním dětském táboře, následně v průběhu měsíce září 2017 žáci 4. a 7. třídy základní školy, kteří se samotného pedagogického experimentu neúčastnili. Celkem test vyzkoušelo 29 respondentů. Ve všech případech měli za úkol nejenom zodpovědět během 50 minut co nejvíce otázek, ale také zapsat přibližný čas, který každé úloze věnovaly, a subjektivní dojem, zda se jim úloha jevila jako lehká, středně těžká či těžká. Z tohoto důvodu byl čas při zkušebním testování prodloužen oproti předpokládanému ostrému testování o 5 minut.

Po vyplnění testu měly děti také za úkol označit v textu ty odstavce, věty či termíny, které pro ně byly méně jasné či dokonce nesrozumitelné. Díky tomu byly odhaleny nejenom chyby, ale i pojmy, které by některým žákům znesnadňovaly samotné vypracování testu a mohly by negativně ovlivnit výsledky. Příkladem může být slovo šálek, použité v jedné z úloh. Ukázalo se, že pro více než dvě třetiny dětí je slovo „šálek“ výrazem neznámým, a bylo proto nahrazeno běžnějším termínem hrnek.

Tabulka 8 – Zkušební testování – obtížnost otázek

	Úloha č.																	
	1	2	3	4	5	6a	6b	7	8	9	10	11	12	13	14	15a	15b	16	Bodů
Respondent 1	L	L	L	L	S	N	L	L	S	L	L	S	S	S	S	S	S	S	11
Respondent 2	T	L	S	S	T	N	L	S	T	N	T	T	S	T	S	T	S	T	4
Respondent 3	L	S	T	S	L	S	L	S	L	L	L	T	S	T	L	T	T	S	5
Respondent 4	L	L	L	S	L	S	L	S	L	T	S	L	L	T	T	T	T	N	11
Respondent 5	L	T	T	S	L	N	S	S	S	N	L	N	L	N	N	N	N	N	4
Respondent 6	L	S	S	L	S	S	L	S	S	T	T	T	S	S	T	T	T	L	5
Respondent 7	S	L	L	L	L	L	N	N	S	N	L	N	S	T	L	L	T	L	9
Respondent 8	S	L	L	L	L	S	N	L	L	T	S	T	N	N	N	T	T	L	8
Respondent 9	L	L	L	L	L	L	L	S	L	L	L	L	L	L	L	L	L	S	18
Respondent 10	L	L	L	L	L	L	N	S	L	L	L	L	L	L	L	L	L	L	13
Respondent 11	L	L	S	L	L	L	L	L	S	L	T	T	S	S	S	T	N	N	10
Respondent 12	T	L	N	T	S	S	N	N	T	N	T	T	L	T	N	N	N	S	5
Respondent 13	L	S	N	L	S	S	S	S	S	S	S	T	S	S	S	S	S	L	8
Respondent 14	L	L	L	L	S	N	L	L	S	L	L	N	S	T	S	T	T	S	14
Respondent 15	L	S	S	L	L	L	L	S	L	T	T	N	L	S	L	T	T	S	10
Respondent 16	T	T	N	T	N	T	T	S	L	S	N	S	S	S	S	S	S	S	5
Respondent 17	L	S	S	N	S	L	L	L	L	S	T	T	L	T	N	N	N	N	10
Respondent 18	L	S	L	S	T	L	N	N	L	T	N	N	N	T	S	T	S	S	6
Respondent 19	L	S	S	L	S	S	L	S	S	T	T	T	S	S	T	N	N	N	11
Respondent 20	L	L	N	N	L	L	N	L	L	S	T	S	L	T	T	T	T	L	9
Respondent 21	S	T	L	N	S	L	N	N	N	S	T	S	S	T	L	N	N	L	6
Respondent 22	L	S	S	L	S	L	N	S	S	L	S	T	S	T	L	T	T	L	9
Respondent 23	L	L	S	S	S	S	S	T	L	L	L	L	L	T	N	N	N	N	5
Respondent 24	L	N	N	N	N	N	N	N	N	N	S	N	S	N	S	N	N	N	4
Respondent 25	S	S	T	T	T	S	T	T	S	S	T	T	T	S	L	T	T	L	3
Respondent 26	L	S	N	N	S	L	S	S	L	T	N	S	S	T	S	T	S	S	12
Respondent 27	L	S	N	N	N	S	N	S	S	T	N	N	T	N	N	N	N	L	5
Respondent 28	L	S	L	N	S	S	L	N	L	S	T	T	S	T	S	S	N	L	10
Respondent 29	L	L	S	S	L	L	L	S	L	S	S	L	L	N	S	T	S	N	12
Správných odp.	23	14	5	9	21	21	11	12	21	15	18	10	13	10	16	5	6	12	
Odhad dětí	L	L/S	L/S	L	L/S	L/S	L	S	L	S	T	T	S	T	S	T	T	L	
Odhad autora	L	S	S	L	S	L	S	S	L	T	T	T	S	T	S	T	T	L	
	Tankování	Vaření čaje	Výroba	Řazení robotů	Áda v bludišti	Montáž Karla	Montáž Karla 2	Šroubky a matičky	Lodička	Barvení DUDU	Šifrování textu	Barvení dlaždic	Místnost s roboty	Výtah	Cesta	Dosazování	Dosazování 2	Hra na schovávanou	
Stanovená obtížnost	L	S	S	S	L	L	S	S	L	S	S	T	S	T	S	T	T	L	

L – Lehká, S – Středně obtížná, T – Těžká, N – nezodpovězeno

Správná odpověď Chybná odpověď Nezodpovězeno

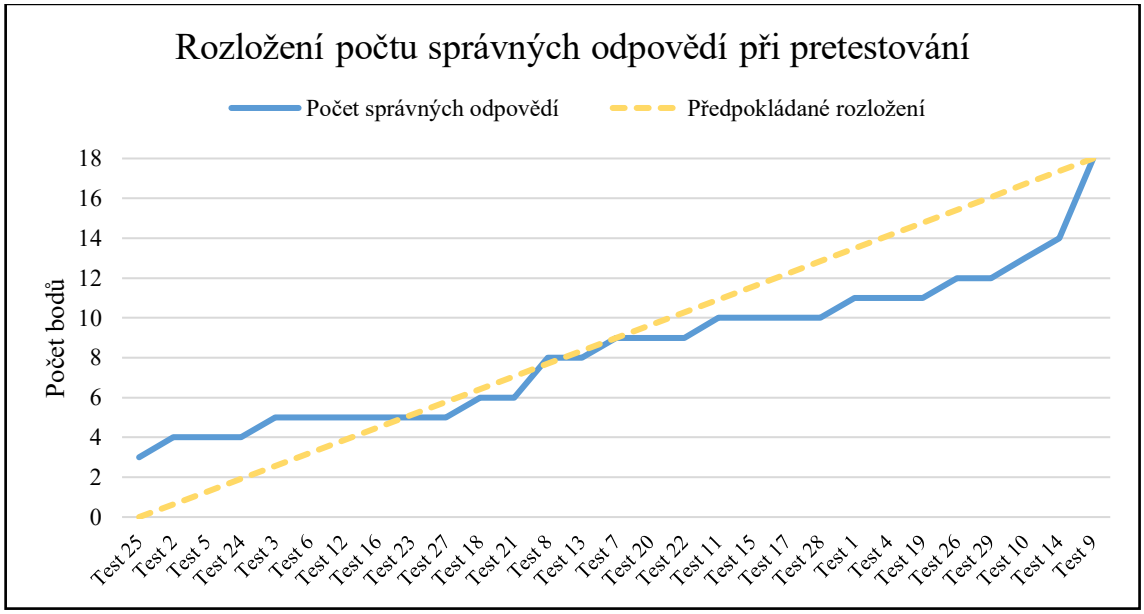
Tabulka 9 – Zkušební testování (čas v minutách)

Úloha č.	1	2	3	4	5	6a	6b	7	8	9	10	11	12	13	14	15a	15b	16	min celkem
Respondent 1	1,5	2	0,8	1	3	N	3	1,5	1	1	2	2	1,5	N	3	2	2	2	29
Respondent 2	3	2	1	1	4	2	2	1	2	N	1	6	2	2	3	3	3	2	40
Respondent 3	2	2	3	2	2	4	2	1	1	1	2	2	2	N	4	N	2	4	36
Respondent 4	1	1	1	2	1	2	1	2	1	3	3	2	1	3	2	3	N	N	29
Respondent 5	1	8	4	2	2	N	N	6	1	N	3	N	3	N	N	N	N	N	30
Respondent 6	1	4	N	2	N	6	N	N	5	10	N	N	10	3	N	N	N	N	41
Respondent 7	N	3	1	1	3	5	1	N	1	N	5	N	5	N	3	2	N	2,5	33
Respondent 8	3	3	1	1	1	8	N	4	1	10	3	5	N	N	N	N	N	N	40
Respondent 9	0,3	1	0,5	0,2	0,8	3	2,5	1	0,5	3,5	2	1,8	1	2,5	1	2,5	2,5	2	29
Respondent 10	1	1	1	1	1	2	N	4	1	1	1	3	2	2	1	1	2	1	26
Respondent 11	0,5	1	1	1	0,5	3	5	1	1	5	5	2	3	3	5	5	N	1	43
Respondent 12	0,5	1	0,5	0,5	1	2	N	1	0,5	1	0,5	1	3,5	1	0,5	1	N	2	18
Respondent 13	1	3	N	2	3	1	3	1	1	4	1	4	1	1	1	3	2	0,5	33
Respondent 14	1	1	1	1	1	1	1	1	1	3	1	N	1	1	1	2	6	1	25
Respondent 15	1	3	2	1	1	1	1	1	1	1	2	N	1	3	1	2	2,5	1	26
Respondent 16	3	2	N	1	N	0,5	2	1	1	5	5	1	7	1	4	1	1	2	38
Respondent 17	1	1	2	N	3	1	1	1	1	3	4	1	1	1	N	N	N	N	21
Respondent 18	1	2,5	2	1	2,5	1	N	N	3	3,5	N	N	N	1	4	1	3	1	27
Respondent 19	4	1	1	1,5	1,5	1	1	N	1	1	1	3,5	1	3,5	1	N	N	N	23
Respondent 20	1	1	N	N	1	1	N	3	1	1	1	5	1	3,5	3,5	1	0,5	1	26
Respondent 21	1	2	0,5	N	1	0,5	N	N	N	5	1	1	1	3	1	N	N	0,5	18
Respondent 22	1	3,5	1,5	1	1,5	1	N	3,5	1	2	1	4	3	5	1	4	4	1	39
Respondent 23	1	2	1	2	1	0,5	1,5	2	1	4	6,5	5	1	1	N	N	N	N	30
Respondent 24	1	N	N	N	N	N	N	N	N	N	1	N	1	N	1	N	N	N	4
Respondent 25	2	4	1	2	1,5	0,5	3	1	1	2,5	1	2,5	2	1	1,5	2	2	1	32
Respondent 26	2	3	0,5	0,5	1	1	1	1	1	1	2,5	1	1	3	1	1	2	2	26
Respondent 27	1	1	N	N	N	1	N	1	1	6	N	N	2	N	N	N	N	1	14
Respondent 28	1	2	1	N	1	1	1	N	1	3	3	1	2	2	1	1	N	1	22
Respondent 29	1	4,5	2,5	2	1,5	1	1	1	0,5	1	2	1	3,5	N	4,5	2	1	N	30
Průměrný čas	1,4	2,3	1,3	1,3	1,6	2,0	1,8	1,8	1,2	3,3	2,3	2,6	2,4	2,2	2,1	2,1	2,4	1,5	36
Stanovený čas	1,5	2,5	1,5	1,5	2	2	2	2,5	1,5	4	2,5	3	3	2,5	2,5	2	2	1,5	40
	Tankování	Vaření čaje	Výroba	Řazení robotů	Áda v bludišti	Montáž Karla	Montáž Karla 2	Šroubky a matičky	Lodička	Barvení DUDU	Šifrování textu	Barvení dlaždic	Místnost s roboty	Výtah	Cesta	Dosazování	Dosazování 2	Hra na schovávanou	

N – nezodpovězeno

Při vyhodnocování zkušebních testů se ukázalo, že nikdo z respondentů nepotřeboval pro vyplnění testu více než 40 minut. Z časů uvedených respondenty při pilotním ověřování byl u každé z úloh určen průměrný čas na každou jednotlivou úlohu. Součet těchto průměrů činil 36 minut, z čehož vyplývá, že respondenti řešili jednu úlohu průměrně 2 minuty. Ukázalo se tedy, že čas běžné vyučovací hodiny je pro vypracování testu dostatečný, a pro test byl tedy stanoven časový limit 45 minut.

Průměrný počet bodů, který respondenti získali při zkušebním testování, činil 8,34 bodu na jeden test z celkového maximálního možného počtu 18 bodů. Nejlepšího výsledku v průběhu zkušebního testování dosáhla žákyně ve věku 13 let, která celý test zvládla bezchybně za 22 minut. Ukázalo se, že v testu se nevyskytly žádné úlohy, které by všechny děti zodpověděly správně, ani úlohy, které by nezodpověděl nikdo.



Graf 12 – Rozložení počtu správných odpovědí při pilotním testování

Na základě výsledků pilotního testování byla vytvořena konečná verze vstupního testu s označením TEST A1 (viz Příloha D). Vstupní TEST A1 obsahuje celkem 18 algoritmických úloh, které mají tři úrovně obtížnosti. Test obsahuje 5 lehkých, 9 středně těžkých a 4 úlohy těžké. Celkový čas na vypracování testu byl stanoven na 45 minut, což činí 2,5 minuty na jednu úlohu. Dvě třetiny úloh testu mají podobu výběru správné odpovědi z nabízených možností. U jedné třetiny úloh se jedná o úlohy s tvořenou odpovědí.

Tabulka 10 – Soupis algoritmických úloh – TEST A1 a TEST A2 (viz Příloha C a Příloha D)

Č.	Název úlohy	Obtížnost	Čas (min)	Popis úlohy	Zaměření úlohy
1	Tankování	Lehká	1:30	Hledání kroků vedoucích k určenému cíli na základě vstupních dat a výstupu algoritmu.	Algoritmizace, logika
2	Vaření čaje	Střední	2:30	Doplnění chybějících částí algoritmu.	Algoritmizace, abstrakce informací, čtenářská gramotnost
3	Výroba	Střední	1:30	Porozumění algoritmu založeného na principu klonování.	Algoritmizace, dekompozice, klonování
4	Řazení robotů	Lehká	1:30	Nalezení nejefektivnějšího postupu řazení.	Algoritmizace (řazení)
5	Áda v bludišti	Lehká	2:00	Tvorba programu dle zadaných parametrů.	Algoritmizace, programování
6a	Montáž Karla	Lehká	2:00	Porozumění algoritmu a doplnění chybějících kroků algoritmu.	Algoritmizace, programování
6b	Montáž Karla 2	Lehká	2:00	Porozumění algoritmu.	Algoritmizace
7	Šroubky, matičky a podložky	Střední	2:30	Nalezení řešení známého algoritmu.	Algoritmizace, logika
8	Lodička	Střední	1:30	Výběr správné varianty algoritmu.	Algoritmizace (řazení), logika
9	Barvení DUDU	Střední	4:00	Stanovení jednotlivých kroků algoritmu na základě vstupních a výstupních dat.	Algoritmizace, logika
10	Šifrování textu	Střední	2:30	Porozumění algoritmu a jeho aplikace.	Algoritmizace, logika
11	Barvení dlaždic	Těžká	3:00	Výběr správné a nejefektivnější varianty algoritmu.	Algoritmizace, programování
12	Místnost s roboty	Těžká	3:00	Pochopení funkce programu. Určení výsledku známého algoritmu.	Algoritmizace, logika
13	Výtah	Těžká	2:30	Výběr správného postupu splňujícího podmínky zadání.	Algoritmizace, logika, programování
14	Cesta	Střední	2:30	Řazení kroků programu dle zadaných podmínek.	Algoritmizace (řazení), logika
15a	Dosazování	Těžká	2:00	Porozumění algoritmu obsahujícího proměnnou hodnotu.	Algoritmizace
15b	Dosazování 2	Těžká	2:00	Porozumění algoritmu obsahujícího proměnnou hodnotu.	Algoritmizace
16	Hra na schovávanou	Lehká	1:30	Vyloučení variant nesplňujících podmínky zadání.	Algoritmizace, logika

4.3 Výuka

Cílem výuky základů programování bylo rozvíjet tvořivé myšlení žáků zaměřené na algoritmizaci. Metodický přístup kombinoval aktivity typu „CS unplugged“ bez počítače s aktivitami na počítači v prostředí Scratch. Výuka byla rozdělena do tří bloků (viz Tabulka 18) a probíhala ve všech skupinách A, B, C a D za stejných kontrolovaných podmínek.

V první části výuky se žáci seznámili s pojmem algoritmus, který byl pro věkovou kategorii 10–12 let v naprosté většině případů termínem zcela neznámým. Dále se žáci naučili orientovat v prostředí programu Scratch. Seznámili se s tříděním příkazů do jednotlivých kategorií. Naučili se otevírat i ukládat projekty, vkládat pozadí a objekty, upravovat je pomocí vestavěného editoru. Seznámili se s vlastnostmi objektů a jejich nastavením.

Ve druhé části výuky se žáci seznámili se základními koncepty programování, kterými jsou sekvence, události, cykly, paralelizace, zprávy, podmínky (If, operátory), proměnné a klonování. Výuka byla rozložena do lekcí, ve kterých žáci upravovali jednoduché hry. Žáci programovali úvodní animace, ovládání postav, upravovali interakce mezi objekty či bodové ohodnocení. Protože hlavním cílem této diplomové práce byl především rozvoj algoritmického myšlení žáků, byly v jednotlivých lekcích přibližovány koncepty programování rovněž pomocí zápisu jednoduchých algoritmů formou vývojových diagramů, které žáci tvořili v součinnosti, za dopomoci učitele a diskusí žáků nad možnými řešeními.

Ve třetí části žáci vytvářeli vlastní hru v prostředí Scratch. Navrhli její grafický design a projekt principu fungování hry na základě interakce objektů. Sestavili vlastní kód z konceptů programování, které si již osvojili. Při této činnosti se seznámili s tvorbou bloků (podprogramů), testováním a laděním programu. Cílem této části pedagogického experimentu v prostředí Scratch bylo upevnění již osvojených znalostí a také motivování žáků k dalším experimentům s programováním i po skončení pedagogického experimentu.

4.3.1 Lekce 1 – Algoritmus

V první lekci se žáci seznámí s pojmem algoritmus a jeho vlastnostmi a s druhy zápisu algoritmů pomocí vývojových diagramů.

Cíl výuky

- Žák dokáže vyjmenovat alespoň pět běžných zařízení, která se neobejdou bez programování.
- Žák dokáže uvést životní situace, které probíhají podle jasně definovaných kroků.
- Žák dokáže popsat jednotlivé kroky konkrétního postupu (vaření čaje, skládání origami apod.).
- Žák vysvětlí základní vlastnosti algoritmu (hromadnost, jednoznačnost, obecnost, konečnost).
- Žák dokáže porozumět jednoduchému zápisu algoritmu, určí základní grafické prvky vývojového diagramu (začátek, konec, zpracování a rozhodování).

Postup práce

(1) Žáci vyplní první úkol z pracovního listu č. 1 (viz Příloha B)
(2) Vysvětlení pojmu algoritmus a jeho vlastností.
(3) Diskuse s žáky – Splňují vaše postupy z pracovního listu podmínky algoritmu?
(4) Vysvětlení pojmu vývojový diagram.
(5) Společná tvorba algoritmu pomocí návodných otázek.

Poznámky pro učitele

Cílem lekce je představit žákům pojem algoritmus tak, aby pochopili, že se s algoritmy setkáváme také v běžném životě a v různých oborech lidské činnosti, a že nesouvisí pouze s prací na počítači.

S algoritmy se setkáváme denně při mnoha činnostech, aniž bychom si to uvědomovali. Často provádíme ustálené postupy, které bychom jen obtížně měnili. Například čištění zubů, jízda výtahem, ranní cesta do školy či ořezávání tužky jdou vykonat různými způsoby, ale vždy obsahují posloupnost kroků, kterou nelze měnit, abychom se dopracovali cíle. Příkladem algoritmu je například jakýkoli kuchařský recept. Algoritmy jsou přesné návody či popis pracovních postupů, kterými lze řešit obdobné typy úloh.

Vlastnosti algoritmu

Hromadnost: Algoritmus není určen pro jednu situaci.

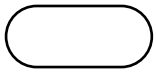
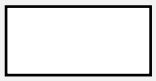
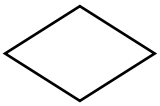
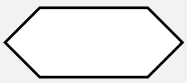
Determinovanost (jednoznačnost): Algoritmus nesmí obsahovat nejednoznačné příkazy jako slova brzy, hodně, více apod. Větu „počítačový virus smazal antivirový program“ lze chápat dvěma způsoby. V prvním případě byl smazán vir, v druhém antivirový program.

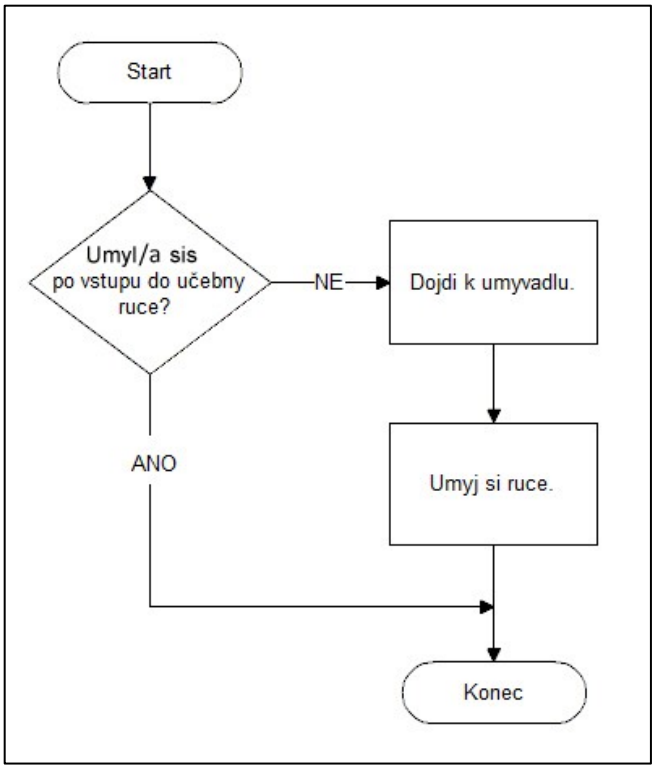
Opakovatelnost: Pokud jsou stejné vstupní hodnoty, musí dojít ke stejnému výsledku.

Rezultativnost (konečnost): Algoritmus musí po vykonání programu skončit.

Algoritmy lze zapisovat verbálně – běžným jazykem (kuchařské recepty, montážní návody), algoritmickým jazykem, programovacím jazykem, nebo graficky (obrázkové návody, origami), vývojové diagramy.

Tabulka 11 – Základní prvky vývojového diagramu

	Konec a začátek algoritmu
	Běžný příkaz
	Podmíněný výraz
	Příprava dat (cyklus s počtem opakování)
	Spojovací čára



Obrázek 6 – Ukázka možného algoritmu

4.3.2 Lekce 2 – Seznámení s programem

V první lekci věnované jazyku Scratch se žáci seznámí s on-line vývojovým prostředím programu Scratch (lze použít i off-line editor), které se skládá ze scény, prostoru pro postavy, galerie příkazů a prostoru pro programování pomocí příkazů. Naučí se vkládat nové objekty do scény a editovat je.

Vstupní znalosti

- Žáci ovládají základy práce s počítačem.

Cíl výuky

- Žák se orientuje na stránce https://Scratch.mit.edu.
- Žák si osvojí práci v prostředí programu Scratch – dokáže otevřít a uložit projekt.
- Žák popíše rozložení oken programu.
- Žák určí, kde se nacházejí základní funkce programu.
- Žák dokáže vložit a editovat nové pozadí.
- Žák dokáže vložit a editovat novou postavu.

Postup práce

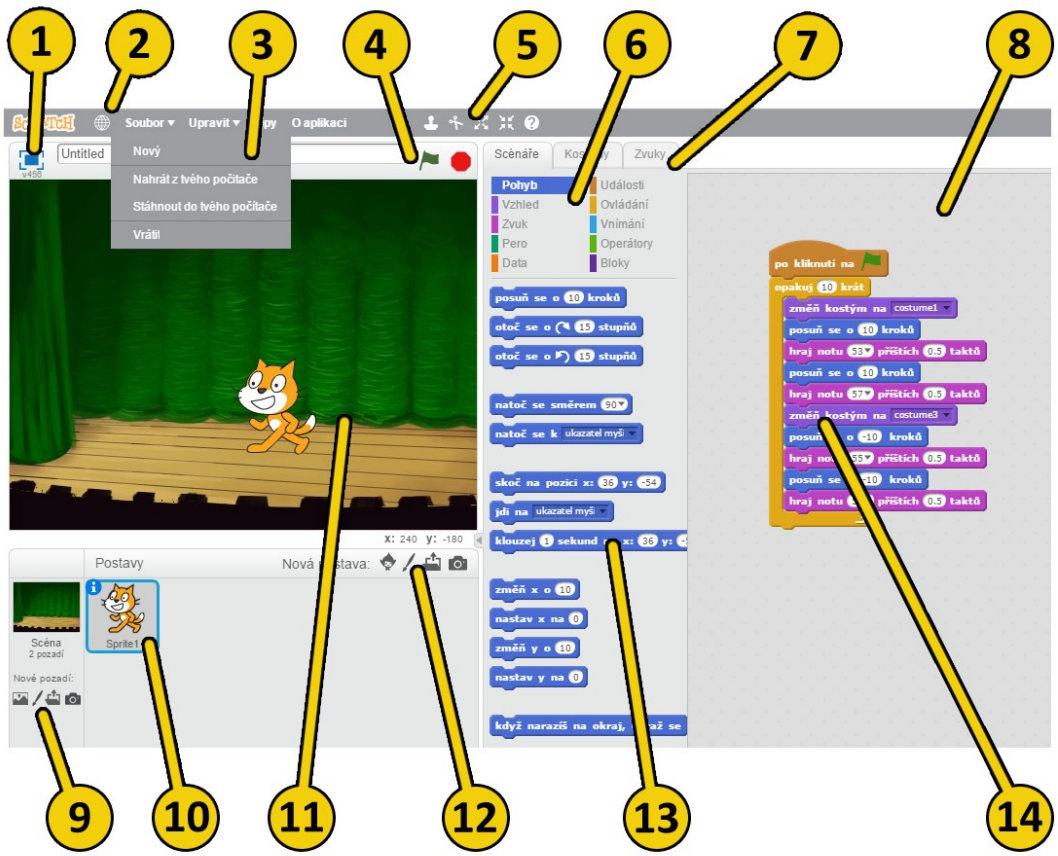
(1) Žák spustí webový prohlížeč (Google Chrome, Mozzilla Firefox apod.).
(2) Žák přejde na stránku webu Scratch – https://Scratch.mit.edu.
(3) Žák spustí editor – záložka tvořit.
(4) Učitel seznámí žáky s prostředím programu Scratch (Obrázek 7).
(5) Učitel seznámí žáky s kategoriemi příkazů (Tabulka 12).
(6) Žák vloží nové pozadí scény z knihovny či souboru.
(7) Žák vloží novou postavu z knihovny či souboru.
(8) Žák upraví postavu a pozadí pomocí nástrojů v editoru (Obrázek 8)

Poznámky pro učitele

Pro usnadnění práce s programem se žáky ve třídě lze doporučit, aby si žáci vytvořili uživatelské účty na portálu https://Scratch.mit.edu. Pro vytvoření účtu je potřeba funkční emailová adresa.

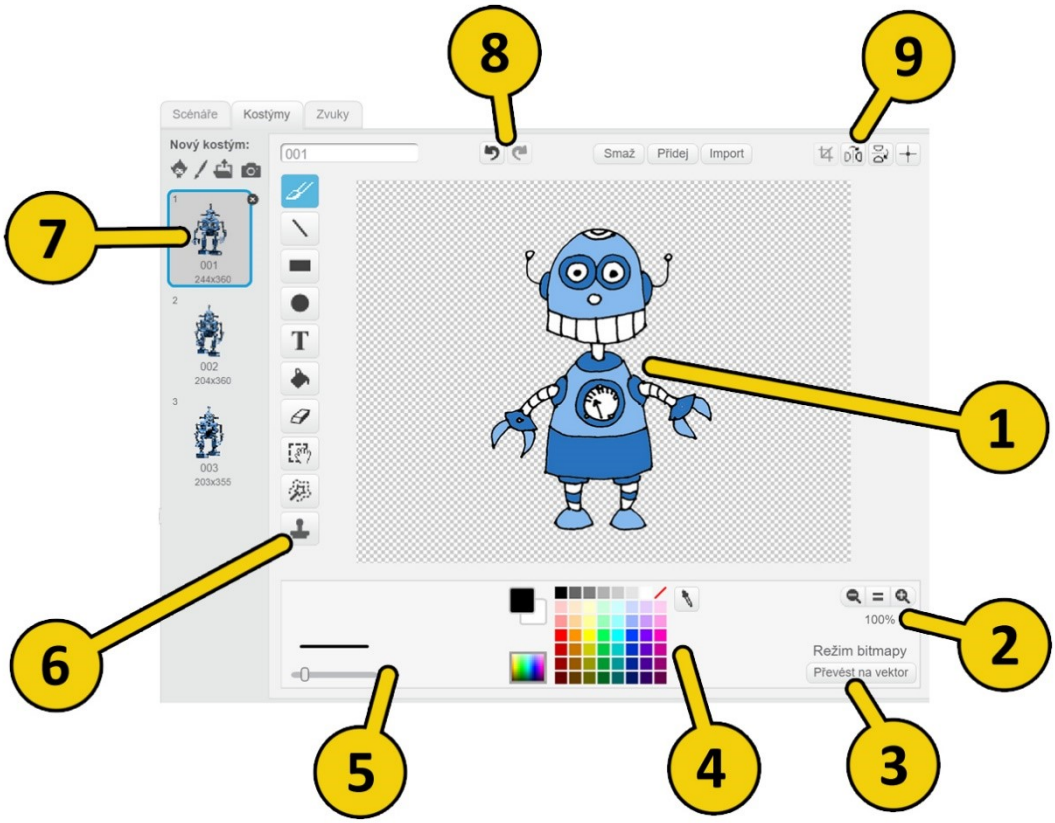
Pro případ nedostupnosti on-line připojení k portálu Scratch.mit.edu je vhodné mít na pracovních stanicích k dispozici off-line verzi programu. Soubory ke stažení potřebné pro instalaci editoru a tipy pro instalaci jsou dostupné z https://Scratch.mit.edu/download.

Výchozí scéna programu Scratch má 480x360 obrazových bodů. Každý bod na scéně lze tedy popsat pomocí souřadnic X (-240;240) a Y (-180;180), jejichž průsečík se nachází uprostřed scény.



1. Maximalizace scény
2. Výběr jazyka
3. Nabídka uložit – otevřít
4. Spustit – zastavit program
5. Nástroje pro úpravu postavy
6. Kategorie příkazů
7. Záložky scénář, kostýmy, zvuky
8. Prostor pro vytváření scénářů
9. Nástroje pro editaci scény
10. Sprite – postava
11. Scéna
12. Nástroje pro editaci postavy
13. Příkazy
14. Programovaný kód

Obrázek 7 – Popis prostředí programu Scratch



1. Editovaný kostým
2. Zvětšení a zmenšení zobrazení
3. Volba druhu grafiky (vektor / bitmapa)
4. Paleta barev
5. Posuvník – tloušťka čáry
6. Kreslicí nástroje
7. Zvolený kostým
8. Krok zpět / vpřed
9. Převrácení, ořez

Obrázek 8 – Grafický editor Scratch

Programovací bloky jsou rozděleny do deseti barevně odlišených skupin: Pohyb, Vzhled, Zvuk, Pero, Data, Události, Ovládání, Vnímání, Operátory a Bloky. Při prvním seznámení s příkazy rozdělenými do kategorií postačí zevrubné seznámení s nejdůležitějšími příkazy každé kategorie.

Tabulka 12 – Kategorie programu Scratch (zdroj: https://cs.wikipedia.org/wiki/Scratch)

Kategorie	Poznámka	Kategorie	Poznámka
Pohyb	Pohyby a natáčení postav	Události	Počáteční bloky určující spouštěcí událost pro navazující posloupnost bloků
Vzhled	Bloky ovlivňující vzhled postav, změny jejich velikosti, komiksové bubliny s řečí nebo myšlenkami, změny pozadí	Ovládání	Podmíněný příkaz když-tak-jinak, cykly a zastavení programu
Zvuk	Přehrávání zvukových souborů a programovatelných nástrojů a tónů	Vnímání	Postavy mohou reagovat na kontakt s okolím, které uživatel vytvořil (např. detekce barev)
Pero	Postava může na scénu kreslit, jakoby držela pero s volitelnou tloušťkou, barvou a intenzitou	Operátory	Aritmetické a logické operátory, základní matematické funkce, náhodná čísla
Data	Vytváření proměnných a práce s jejich hodnotami	Bloky	Uživatelské procedury (bloky) a řízení připojených externích zařízení (třeba robotů)

4.3.3 Lekce 3 – Vlastnosti objektu

Objekty v programování podobně jako v reálném životě popisujeme konkrétními vlastnostmi. V případě programovacího jazyka Scratch lze za objekty považovat scénu a postavu (angl. sprite). Základními vlastnostmi postav jsou velikost, pozice v souřadnicovém systému, viditelnost, zvuk, orientace a styl otáčení. Objekty do prostředí editoru lze buď vybrat a vložit z knihovny, vytvořit v editoru, nahrát ze souboru nebo importovat pomocí připojené kamery.

1. Název postavy
2. Směr pohybu postavy
3. Styl otáčení postavy
4. Přemístitelnost hráčem
5. Viditelnost
6. Náhled postavy
7. Návrat

Obrázek 9 – Vlastnosti objektu

Cíl výuky

- Žák dokáže vyjmenovat vlastnosti objektu v programu Scratch.
- Žák dokáže nastavit vlastnosti objektu v programu Scratch.

Postup práce

(1) Učitel s žáky diskutuje na téma vlastnosti objektů, na tabuli sepíše názvy kategorií vlastností.

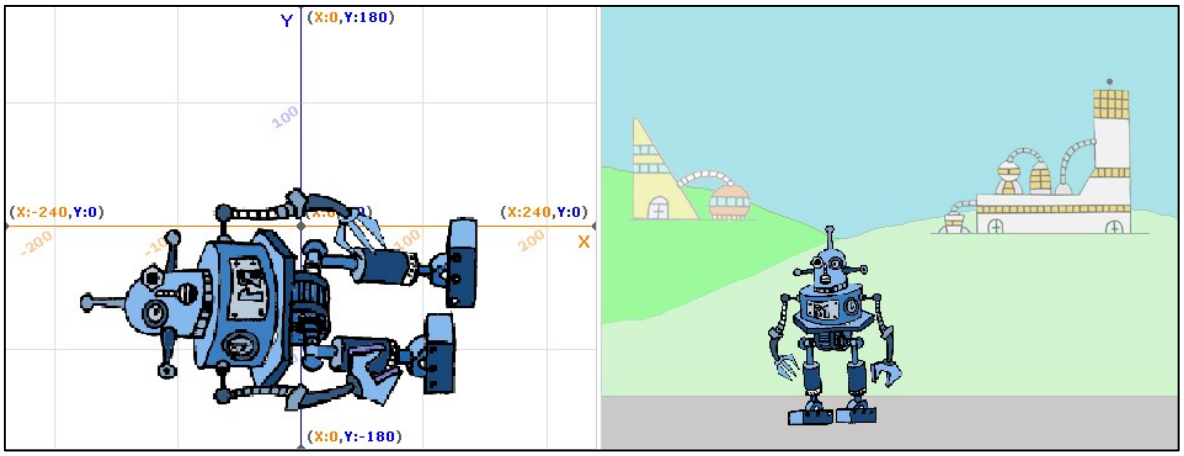
(2) Žáci otevřou webový prohlížeč a přejdou na adresu webu Scratch (https://Scratch.mit.edu, záložka Tvořit).

(3) Žáci otevřou připravený projekt Vlastnosti-zadání s pozadím souřadnicového systému (https://scratch.mit.edu/projects/228405319/).

Žáci mají za úkol upravit:

- název postavy (jméno robota)
- velikost postavy (zmenšit)
- směr postavy
- styl otáčení
- pozici v souřadnicovém systému
- viditelnost a přemístitelnost hráčem.

(4) Společná kontrola výsledků.



Obrázek 10 – Zadání a výsledná podoba cvičení Vlastnosti

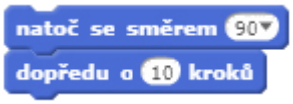
Poznámky pro učitele

Cílem cvičení je seznámit žáky s vlastnostmi postav (objektů) ve Scratch. V projektu se nachází postava robota, u které je třeba nastavit vlastnosti. Program obsahuje sadu příkazů pro pohyb robota po ploše programu, animaci jeho chůze a zvuky. Konečná podoba žákovských prací by měla obsahovat zmenšenou postavu robota, který přechází vodorovně po ploše v úrovni Y = -70 (viz https://scratch.mit.edu/projects/228405380/).

4.3.4 Lekce 4 – Sekvence

V této lekci se žáci seznámí se s konceptem sekvence. Úkolem žáků je sestavit sekvenci příkazů, podle kterých se postava robota přesune z původní pozice na spodní okraj scény a přitom kopíruje klikatou cestu. Je třeba využít znalosti z předchozí lekce a pomocí příkazů mu nastavit viditelnost, velikost, pozici v souřadnicovém systému a kostým. Žáci, kteří jsou při práci rychlejší, mohou použít scénu s čistým nebem, zobrazit postavy mraků a nechat je volně plout po obloze. Příkazy kategorie Pohyb umožňující pohyb postav po scéně jsou zobrazeny v tabulce č. 13.

Tabulka 13 – Příkazy kategorie Pohyb ve Scratch

Pro pohyb obrázku robota po scéně lze použít příkazy změň souřadnice.	
Druhou možností je natočit objekt konkrétním směrem a pozici změnit příkazem.	
Nejvhodnější je využít příkaz „klouzej“. Počet příkazů se tím výrazně sníží.	

Cíl výuky

- Žák dokáže vysvětlit pojem sekvence a nutnost pořadí konkrétních příkazů.
- Žák dokáže vytvořit funkční sekvenci příkazů na základě události „Po spuštění programu“.
- Žák dokáže pomocí příkazů měnit pozici objektu v souřadnicovém systému.

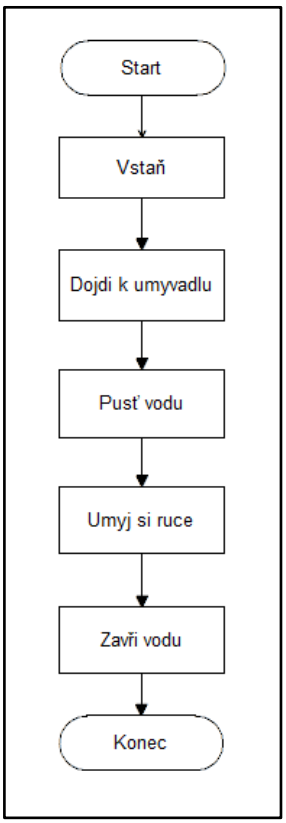
Postup práce

(1) Učitel vysvětlí pojem sekvence (řada, posloupnost, následnost jednoho kroku za druhým).

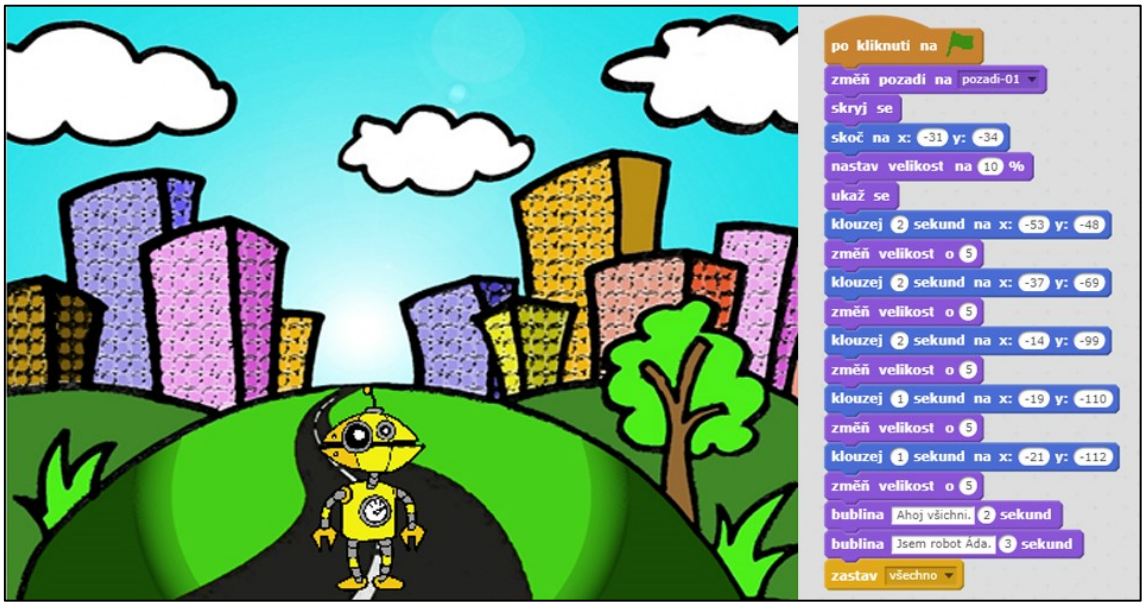
(2) Žáci mají za úkol ve dvojicích vymyslet a obhájit situace, kdy není možné měnit sled kroků (vaření čaje, cesta do školy, instalace hry).

(3) Žáci sestaví algoritmus skládající se pouze ze základních příkazů (viz Obrázek 11).

(4) Učitel předvede ukázkový soubor Sekvence-finish bez zobrazení kódu (https://scratch.mit.edu/projects/228418971/).
(5) Žáci otevřou projekt Sekvence-zadání (https://scratch.mit.edu/projects/228409476/).
(6) Žáci navrhnou slovy, jak by výsledná animace měla vypadat (robot se objeví na horizontu, robot přejde po klikaté cestě dopředu, robot pozdraví).
(7) Úkolem žáků je sestavit sekvenci příkazů, podle kterých se postava robota přesune z původní pozice na spodní okraj scény a přitom kopíruje klikatou cestu. Na konci animace se musí zobrazit bublina s textem pozdravu.
(8) Společná kontrola výsledků.



Obrázek 11 – Algoritmus – sekvence



Obrázek 12 – Příklad využití konceptu sekvence

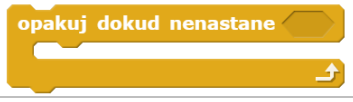
4.3.5 Lekce 5 – Cyklus

V této lekci se žáci seznámí s konceptem cyklu. Cyklus je opakování jednoho příkazu nebo sekvence příkazů. Scratch rozeznává tři druhy cyklů. Cyklus s určeným počtem opakování (opakuj n krát), nekonečný cyklus (opakuj stále) a cyklus s podmínkou na konci (opakuj, dokud nenastane). Cyklus s podmínkou na začátku je třeba sestavit z podmínky a následného příkazu opakuj. V této lekci se žáci seznámí s cyklem s určitým počtem opakování a s cyklem s podmínkou na konci.

Cyklus s pevným počtem opakování (v programování cyklus FOR – cyklus s řídící proměnnou) se provede tolikrát, na jakou hodnotu je nastavena proměnná (n). Při každém provedení se proměnná sníží o jedna (n = n – 1). Pokud hodnota proměnné dosáhne 0, cyklus je ukončen.

Cyklus s podmínkou na konci (v programování cyklus REPEAT) musí proběhnout aspoň jednou. V Scratch se jedná o příkaz „opakuj dokud“. Na konci cyklu příkazů proběhne kontrola podmínky. Pokud je podmínka splněna, cyklus je ukončen. Příkazy kategorie Cyklus jsou zobrazeny v Tabulka 14.

Tabulka 14 – Příkazy Opakuj ve Scratch

Nekonečný cyklus	
Cyklus s pevným počtem opakování	
Cyklus s podmínkou na konci	

V této lekci žáci pracují s upraveným projektem z předchozí lekce. Úkolem žáků je animovat postavu robota pomocí příkazů střídajících kostýmy. Vytvořit cyklus s určeným počtem opakování.

Cíl výuky

- Žák dokáže vysvětlit pojem cyklus.
- Žák dokáže vytvořit ve scénáři funkční cyklus s určeným počtem opakování.
- Žák dokáže použít příkaz cyklus pro animaci postavy pomocí střídání kostýmů.

Postup práce

(1) Učitel představí pojem cyklus (cyklus s počtem opakování). Úkolem žáků je najít situace z běžného života, při kterých dochází k opakování stejné situace (chůze, čištění zubů, hraní karet apod.).

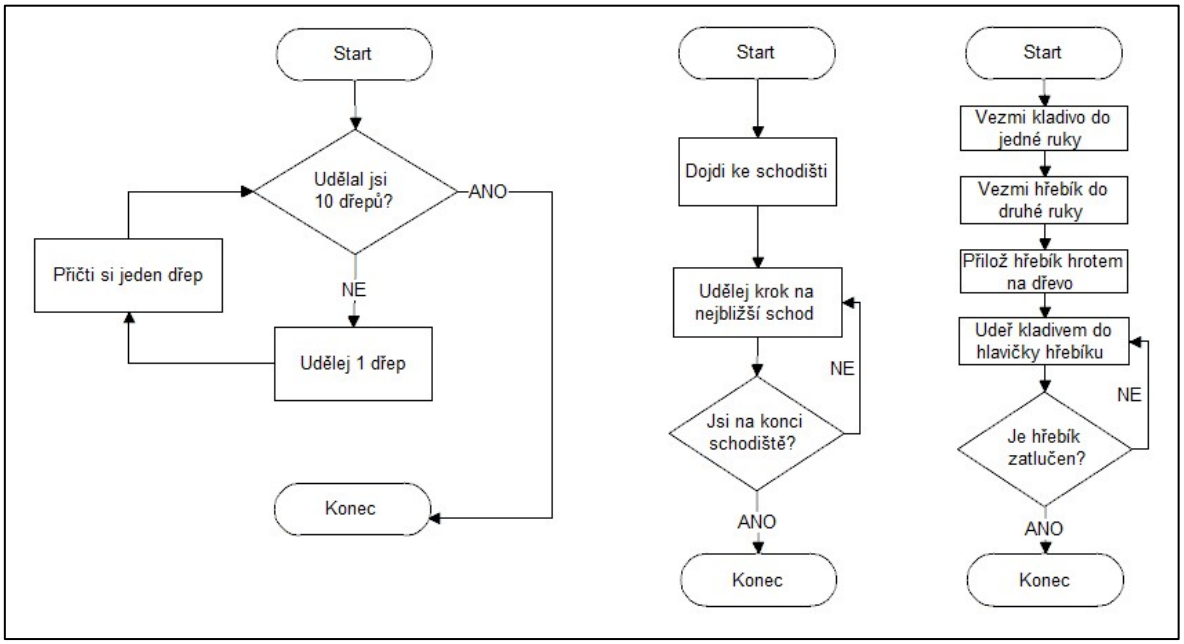
(2) Společná tvorba vybraného algoritmu na tabuli. Učitel s žáky vytvoří jednoduchý algoritmus simulující jednoduchý cyklus (Obrázek 13).

(3) Žáci otevřou projekt Cyklus-zadání (https://scratch.mit.edu/projects/228536310/), cílem práce je animace chůze robota.

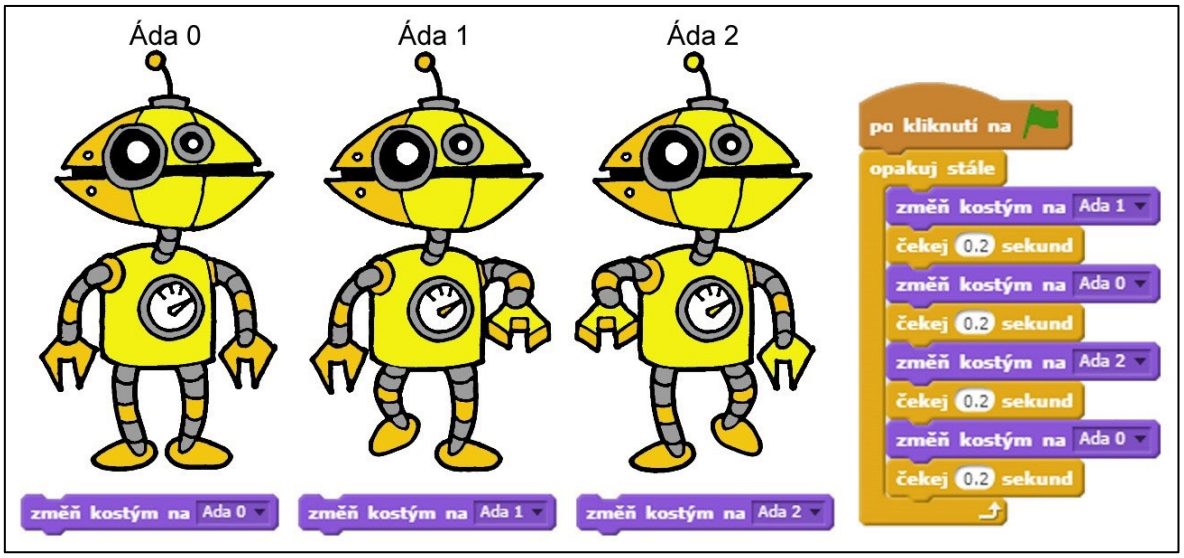
(4) Žáci dostanou za úkol sestavit cyklus střídání kostýmů postavy robota tak, aby působilo dojmem chůze. Druhý script musí postupně zvětšovat postavu robota, aby byl vyvolán dojem jeho přibližování.

(5) Rychlejší žáci mohou podobným způsobem animovat mraky na obloze.

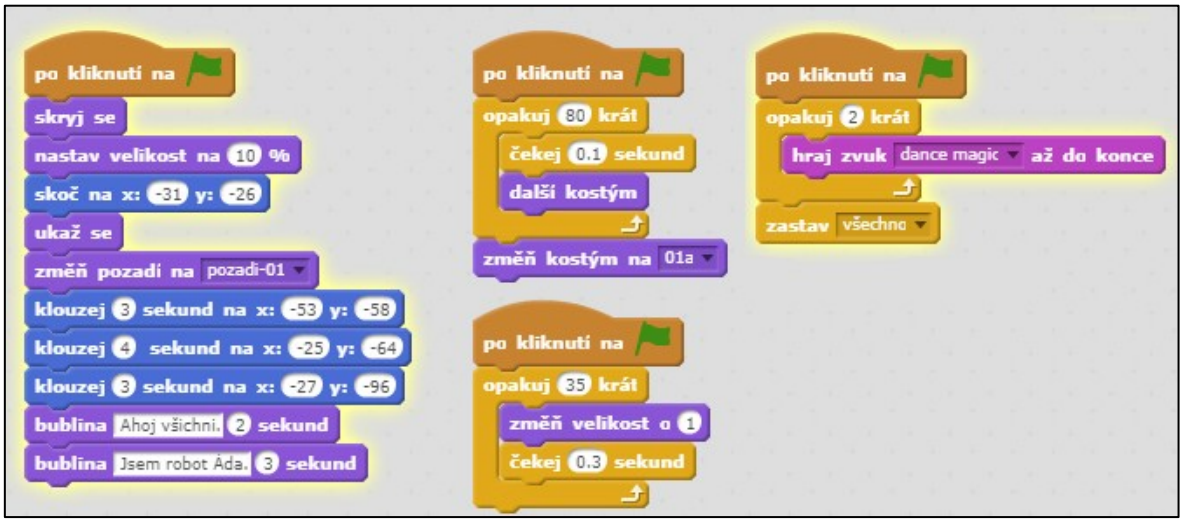
(6) Společná kontrola výsledků.



Obrázek 13 – Příklady algoritmů s konceptem Cyklus



Obrázek 14 – Příklad použití konceptu Cyklus



Obrázek 15 – Ukázka možného řešení úlohy Cyklus

Poznámky pro učitele

Do sekvence příkazů je třeba přidat časovou prodlevu, jinak střídání kostýmů probíhá tak rychle, že jej nelze sledovat jako chůzi.

Většina žáků při prvních pokusech animuje obrázky chybně v pořadí kostým 0 → kostým 1 → kostým 2.

Žáci často využívají místo příkazu změň kostým na „název kostýmu“ příkaz další kostým. To vede opět k chybnému střídání kostýmů. Je vhodné s žáky diskutovat, zda neexistuje řešení, při kterém lze použít příkaz „Další kostým“. Tímto řešením je opětovné vložení kostýmu č. 0 (obě nohy robota na zemi) a seřazení všech čtyř kostýmů v pořadí 1-0-2-0. Tím současně dosáhneme toho, že při skončení animace bude robot stát oběma nohama na zemi.

Ukázka možného řešení úlohy na https://scratch.mit.edu/projects/228535119/.

4.3.6 Lekce 6 – Paralelizace

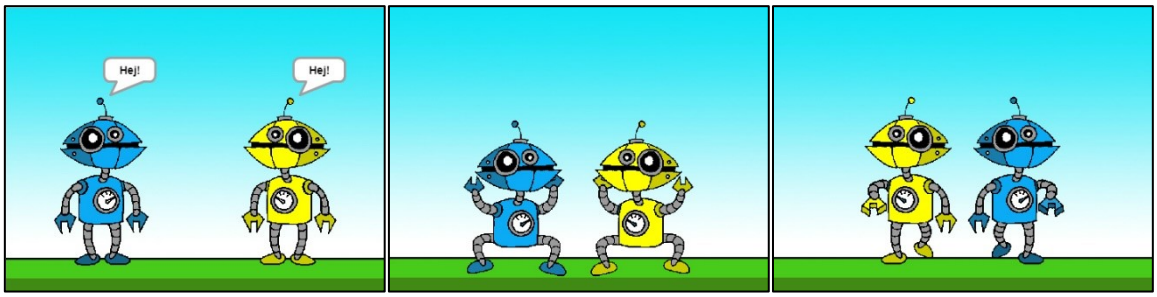
V této lekci se žáci seznámí s konceptem paralelizace. Jedná se o koncept, kdy v případě, že jsou dvě samostatná vlákna programu spuštěna ve stejný okamžik, jsou oba procesy vykonávány souběžně. To umožní například synchronizovat pohyb postav.

Cíl výuky

- Žák dokáže vysvětlit pojem paralelní.
- Žák dokáže vysvětlit funkci a výhody konceptu paralelizace.
- Žák při tvorbě kódu vhodně využívá koncept paralelizace.

Postup práce

(1) Učitel seznámí žáky s pojmem paralelní.
(2) Žáci otevřou projekt Paralelizace-zadání (https://scratch.mit.edu/projects/228419347/)
(3) Žáci vytvoří dvě paralelní vlákna programu pro postavy robotů tak, aby roboti synchronně tančili
(4) Žáci vylepší program přidáním zvuků a dalších objektů
(5) Společná kontrola výsledků.

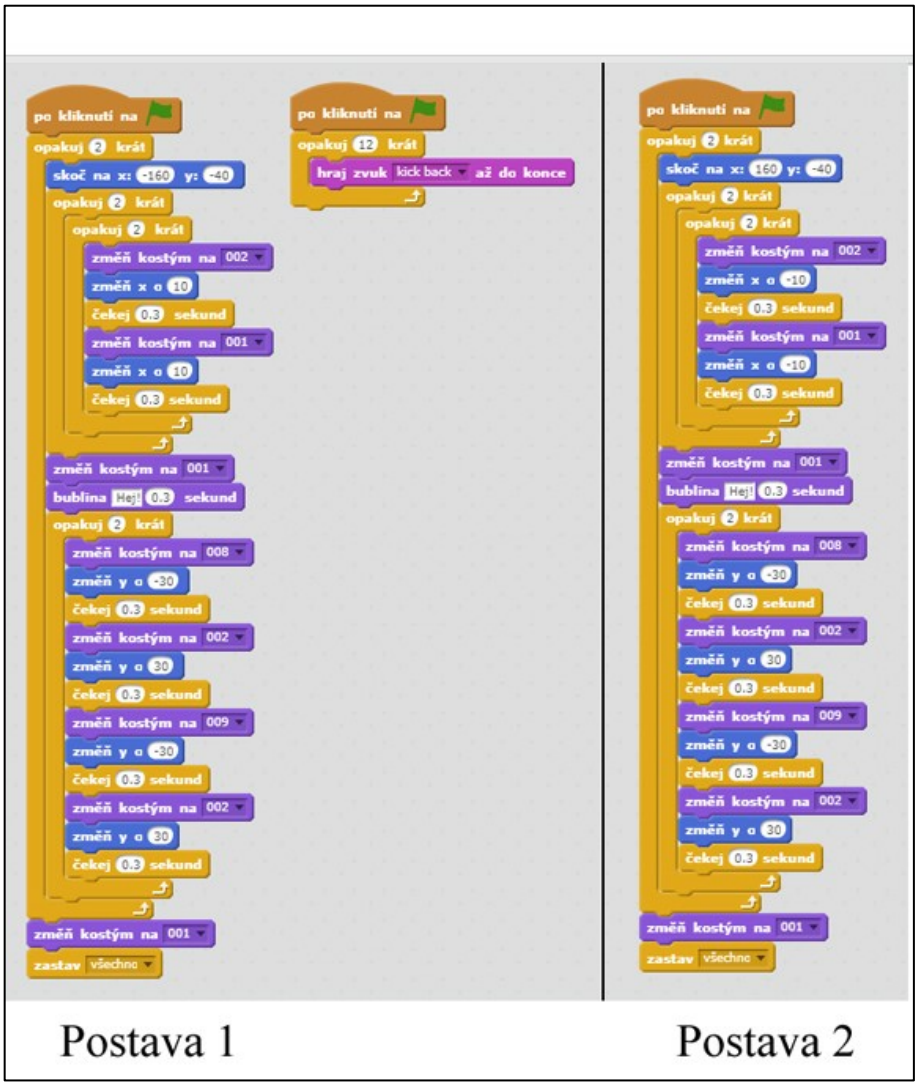


Obrázek 16 – Paralelní tanec robotů

Poznámky pro učitele

Pojem paralelizace je pro žáky ve věku 10–12 let naprosto neznámý. Spojují si jej nejčastěji ve významu paralelních světů, což je pro tuto věkovou kategorii záležitost jen těžko pochopitelná. Pro snazší porozumění principu paralelizace je vhodné žákům tento termín přiblížit názorně. Například lze žákům přehrát videa s paralelním slalomem snowboardingu či lyžování, nebo nechat žáky vykonávat nějakou společnou činnost současně. Nechat je zpívat dvouhlasně či tančit ve dvojicích, kdy každý z partnerů má určeny jiné kroky.

Ukázka možného řešení úlohy na https://scratch.mit.edu/projects/228409668/.



Obrázek 17 – Možné řešení úlohy Paralelizace

4.3.7 Lekce 7 – Události

Doposud žáci vytvářeli krátký scénář, který spouštěli pouze pomocí příkazu „Po kliknutí na start“ nebo poklepáním na jednotlivý příkaz nebo sekvenci příkazů. Aby mohli tvořit složitější programovací koncepty, musí se seznámit s dalšími variantami událostí. Program Scratch pracuje s dvěma kategoriemi událostí. Jedná se o:

a) události zásahu uživatele:
   - po kliknutí na start
   - po stisku určité klávesy

- po kliknutí na mne

b) události v průběhu programu

- po obdržení zprávy
- po změně pozadí na
- když hlasitost (stopky, video) dosáhnou hodnoty větší než stanovená hranice.

Cíl výuky

- Žák dokáže vysvětlit pojem událost.
- Žák dokáže vhodně použít příkazy událostí.

Postup práce

(1) Učitel představí pojem událost.

(2) Společná tvorba vybraného algoritmu na tabuli. Učitel s žáky vytvoří jednoduchý algoritmus simulující jednoduchý cyklus.

(3) Žáci otevřou projekt Události-zadání (https://scratch.mit.edu/projects/228618709/)

(4) Úkolem žáků je naprogramovat pohyb postavy žraloka po ploše scény tak, aby objekt reagoval na stisk kurzorových kláves přesouváním ve směru stisknuté klávesy.

(5) Společná kontrola výsledků.

Tabulka 15 – Příkazy konceptu Události v Scratch

Po kliknutí na lze v programu využít pouze jednou při startu programu. Umožňuje spuštění programu v celoobrazovkovém módu.	
Umožňuje programu reagovat na stisk konkrétní nebo i libovolné klávesy.	
Program reaguje na kliknutí myší na konkrétní objekt. Chceme-li, aby reagoval objekt jiný, kombinujeme s příkazy „rozešli zprávu“ a „po obdržení zprávy“.	
Spustí vlákno programu po obdržení konkrétní zprávy.	
Spustí vlákno programu, pokud dojde ke změně pozadí na pozadí s určitým názvem.	
Spustí vlákno programu v případě, že hlasitost, stopky či přehrávání videa dosáhne zadané hodnoty.	

Úkolem žáků je naprogramovat pohyb postavy po ploše scény tak, aby objekt reagoval na stisk kurzorových kláves přesouváním ve směru stisknuté klávesy. Začneme nejjednodušším způsobem, kdy naprogramujeme vlákno programu pomocí příkazů „po stisku klávesy šipka vpravo“ → „změň souřadnici X o n“.

Obrázek 18 – Příklad použití konceptu Události

Poznámky pro učitele

Rychlejší žáci mohou přidat kód pro spuštění programu či ukončení programu na libovolnou událost, změnu barvy ryby po kliknutí na ni, spuštění hudby po stisku určité klávesy apod.

Ukázka možného řešení úlohy na https://scratch.mit.edu/projects/228616806/.

Plynulejšího pohybu lze dosáhnout použitím konceptu Podmíněný příkaz (viz https://scratch.mit.edu/projects/228616989/). Ten však žáci během této výuky ještě neprobírali.

4.3.8 Lekce 8 – Zprávy

Programování ve Scratch umožňuje vytváření a pojmenování vlastních událostí ve formě zpráv. Ve scénáři lze použít příkaz rozešli zprávu „libovolný-název“ všem, na který jiný scénář po obdržení zprávy „libovolný-název“ může reagovat. Tento příkaz lze využít například pro komunikující postavy, aniž by bylo nutné časovat jednotlivé texty, neboť postavy na sebe mohou navzájem reagovat. Při základech programování zprávu nejčastěji

využijeme při rozhovorech postav a jednodušších animacích. V této lekci mají žáci za úkol vytvořit jednoduchý vtip, ve kterém bude konverzace postav synchronizována pomocí zpráv.

Cíl výuky

- Žák dokáže vysvětlit pojem zpráva z programu Scratch.
- Vhodně používá příkazy událostí „Rozešli zprávu“ a „Po obdržení zprávy“.

Postup práce

(1) Učitel představí žákům koncept zpráv a příkazy „rozešli zprávu“ a „po obdržení zprávy“. Lze využít ukázkový soubor Zprávy-finish (https://scratch.mit.edu/projects/228438636/)

(2) Žáci obdrží karty s vytištěnými sekvencemi příkazů, které postupně vykonávají. (viz obr. Obrázek 21)

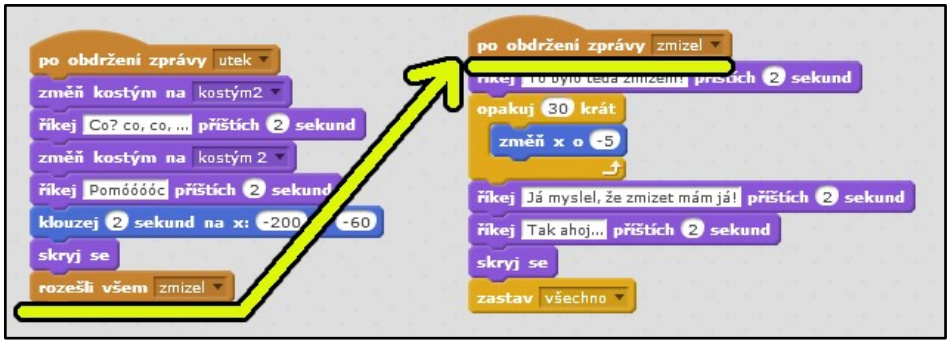
(3) Žáci otevřou webový prohlížeč, zadají adresu webu Scratch (https://Scratch.mit.edu, záložka „tvořit“).

(4) Žáci vloží minimálně dvě postavy.

(5) Žáci vytvoří vtip založený na komunikaci postav. Synchronizace komunikace musí být založena na rozesílání a přijímání zpráv, nikoli na časování.

(6) Prezentace výsledných programů, vzájemné hodnocení.

Obrázek 19 – Příklad použití konceptu Zpráva

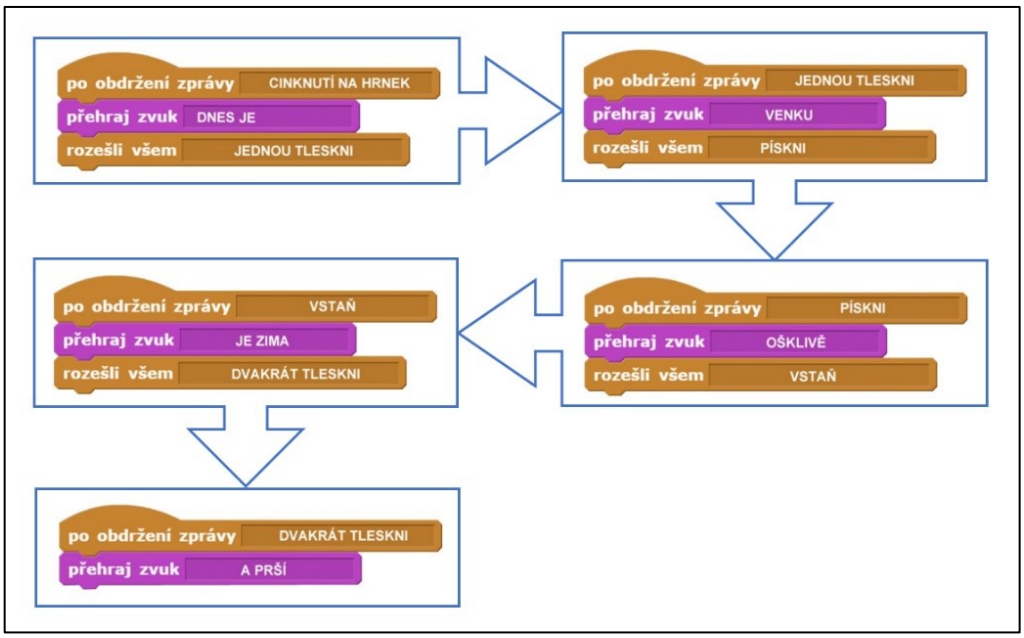


Obrázek 20 – Příklad použití konceptu Zpráva

Poznámky pro učitele

Pro řadu žáků je problém správně pochopit princip rozesílání zpráv ve Scratch, neboť jej zaměňují s příkazem „Říkej“. Jedná se o problém způsobený překladem prostředí Scratch z anglického do českého jazyka. Je proto vhodné hledat s žáky synonyma či víceslovná označení pro jednotlivé příkazy. Osvědčily se termíny signál či pokyn u pojmu zpráva a „Zobraz bublinu s textem“ v případě příkazu „Říkej“. Pokud přesto dojde k situaci, že žáci příkazy zaměňují, je vhodné pro upevnění znalosti zařadit bod (2) na počátku dalších lekcí.

S pomocí karet lze žákům předvést i chybně zapsaný program, kdy některé příkazy neproběhnou, protože sekvence neobdrží příslušnou zprávu, nebo vytvořit nekonečný cyklus.



Obrázek 21 – Karty pro vysvětlení konceptu Zprávy

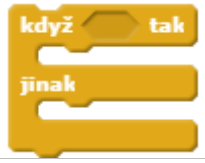
4.3.9 Lekce 9 – Když

V této lekci se žáci seznámí s podmíněnými příkazy. Jedná se o otázky, na které lze odpovědět pouze ANO nebo NE. Podmíněné příkazy dělíme na příkazy úplné a neúplné. V případě úplného podmíněného příkazu se program dělí na dvě části:

1. na vlákno, které proběhne, pokud je podmínka splněna
2. na vlákno, které proběhne v případě, že podmínka neplatí

V programu Scratch se jedná o příkaz „Když-tak-jinak“. Neúplný podmíněný příkaz má pouze vlákno pro případ, že je podmínka splněna, v opačném případě je program ukončen.

Tabulka 16 – Příkazy Když v Scratch

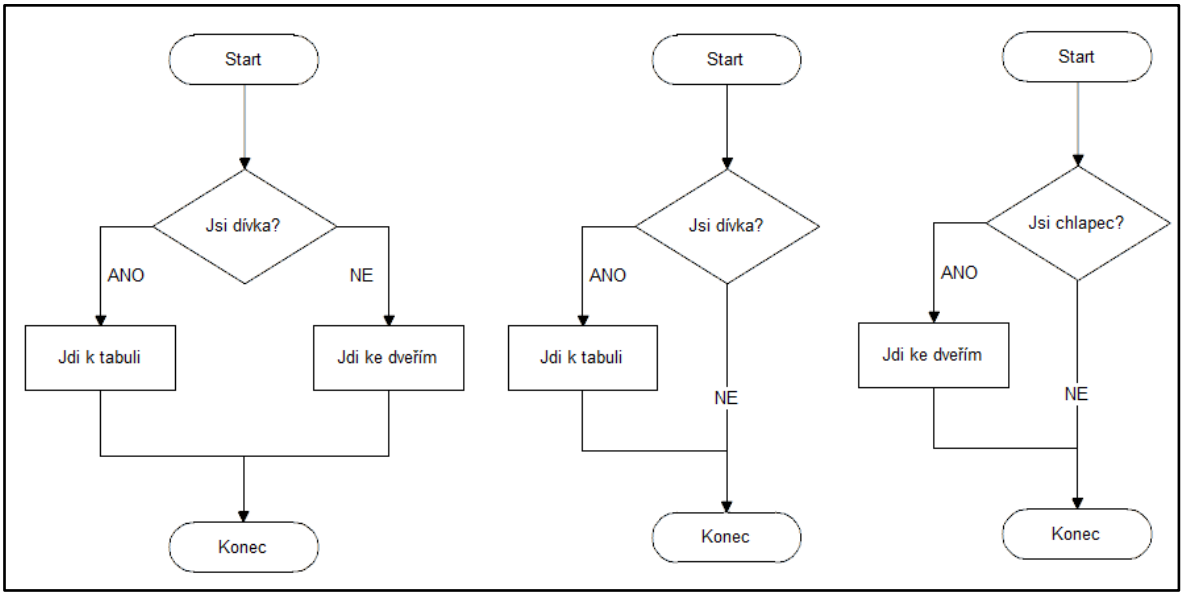
Neúplný podmíněný příkaz. Blok příkazů uvnitř se vykoná pouze v případě, že je podmínka splněna, v opačném případě program pokračuje dále nebo je ukončen.	
Úplný podmíněný příkaz. Program se dělí na dvě části. Na blok příkazů, který proběhne, pokud je podmínka splněna a na blok příkazů, který proběhne v případě, že podmínka neplatí.	

Cíl výuky

- Žák chápe větvení programu.
- Žák dokáže vytvořit ve scénáři podmíněný příkaz pomocí příkazů když-tak, když-tak-jinak.
- Žák dokáže v podmínce použít vnímání a operátory.

Postup práce

(1) Učitel seznámí žáky s pojmem KDYŽ.
(2) Učitel společně s žáky vytvoří algoritmus s podmínkou.
(3) Žáci otevřou projekt Když-zadání (https://scratch.mit.edu/projects/228818652/).
(4) Žáci mají za úkol dokončit hru pomocí konceptu Když.
(5) Společná kontrola výsledků.



Obrázek 22 – Příklad algoritmu s podmínkou

Obrázek 23 – Příklad použití konceptu Když

Poznámky pro učitele

Úkolem žáků je dokončit hru díky využití konceptu Když.

1. Když se dotkne Robot Áda červené barvy, musí změnit kostým na "Leh" a hra se zastaví.
2. Žáci musí vytvořit dojem gravitace. Když se Áda nedotýká země (barvou na botách barvy trávy) musí padat dolů (bude se stále měnit jeho souřadnice Y - mínus).

3. Když se Áda dotkne hvězdy, změní se proměnná hvězdy o +1.
4. Když zmáčkneme klávesu šipka doprava, bude Áda měnit souřadnici X (do plusu), dokud klávesu nepustíme.

Ukázka možného řešení úlohy na https://scratch.mit.edu/projects/228818385/.

4.3.10 Lekce 10 – Proměnná

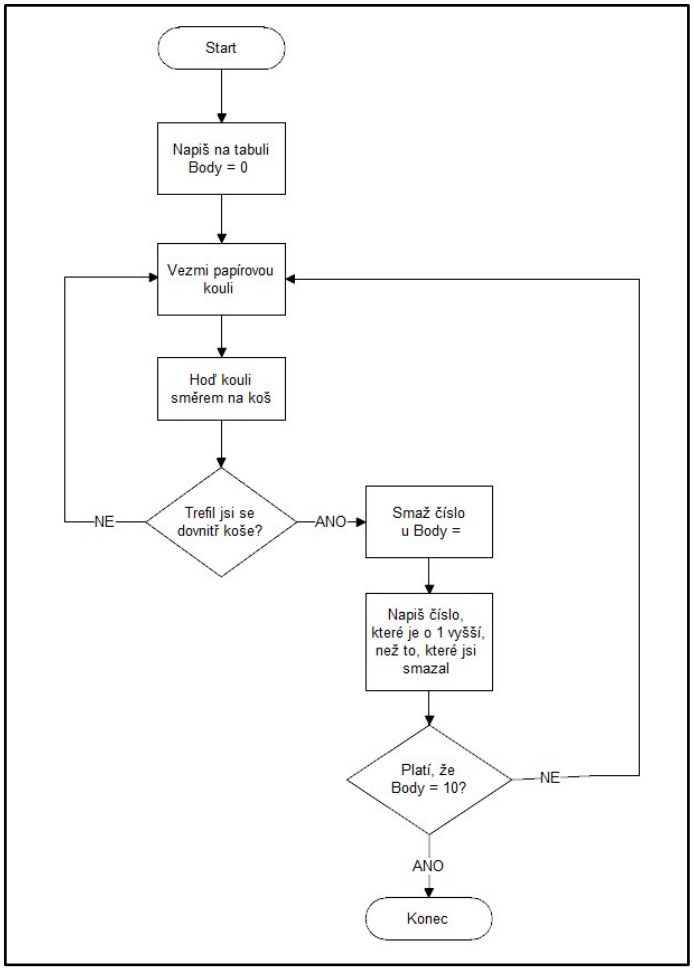
V programování se jedná o symbolické označení paměťového místa, kde je v průběhu programu uchovávána nějaká hodnota, která v průběhu programu může měnit svoji hodnotu. V této lekci se žáci seznámí s pojmem proměnná (označením místa programu), který v našem případě bude zastupovat hodnotu číselnou. Scratch umožňuje vytvořit číselnou proměnnou v sekci data (oranžová kategorie) pod libovolným názvem. Lze používat i českou diakritiku.

Cíl výuky

- Žák dokáže vysvětlit pojem proměnná.
- Žák dokáže vytvořit proměnnou.
- Žák dokáže vytvořit kód, který přiřadí proměnné konkrétní hodnotu.
- Žák dokáže vytvořit kód, který mění hodnotu proměnné.

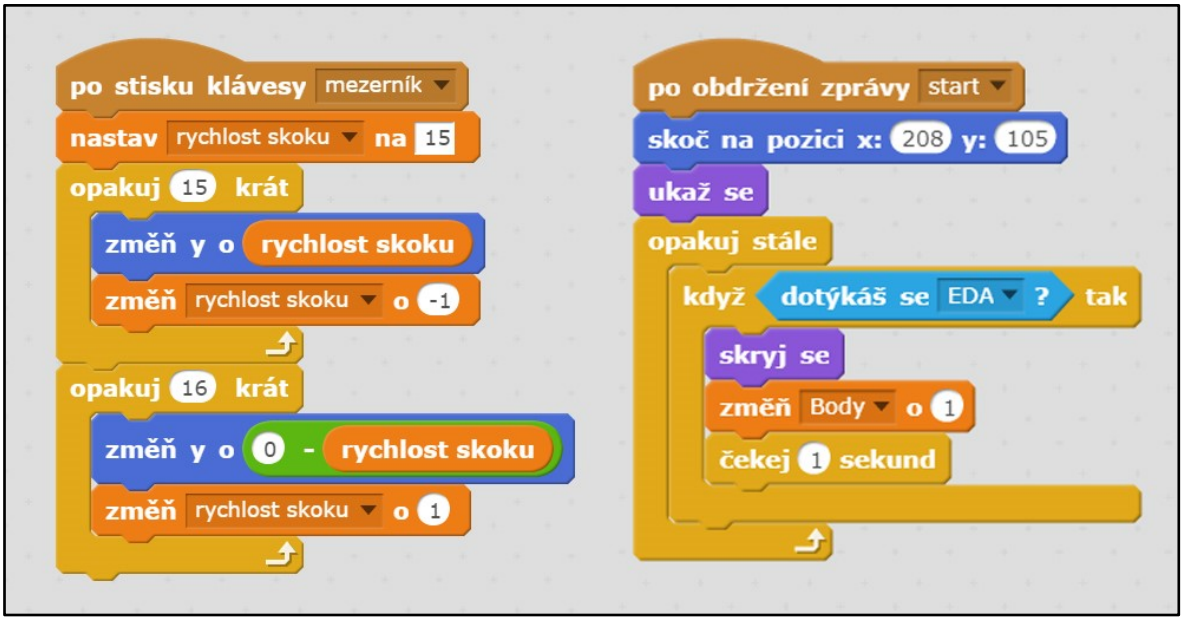
Postup práce

(1) Učitel seznámí žáky s pojmem proměnná.
(2) Učitel společně s žáky vytvoří algoritmus s proměnnou (Obrázek 24).
(3) Žáci otevřou projekt Proměnná-zadání (https://scratch.mit.edu/projects/228634049/)
(4) Žáci vytvoří proměnnou „Mezičas“
(5) Žáci vytvoří kód výskoku pomocí proměnné.
(6) Žáci mohou vytvořit zkrácenou verzi scriptu pomocí rekurze.
(7) Společná kontrola výsledků.

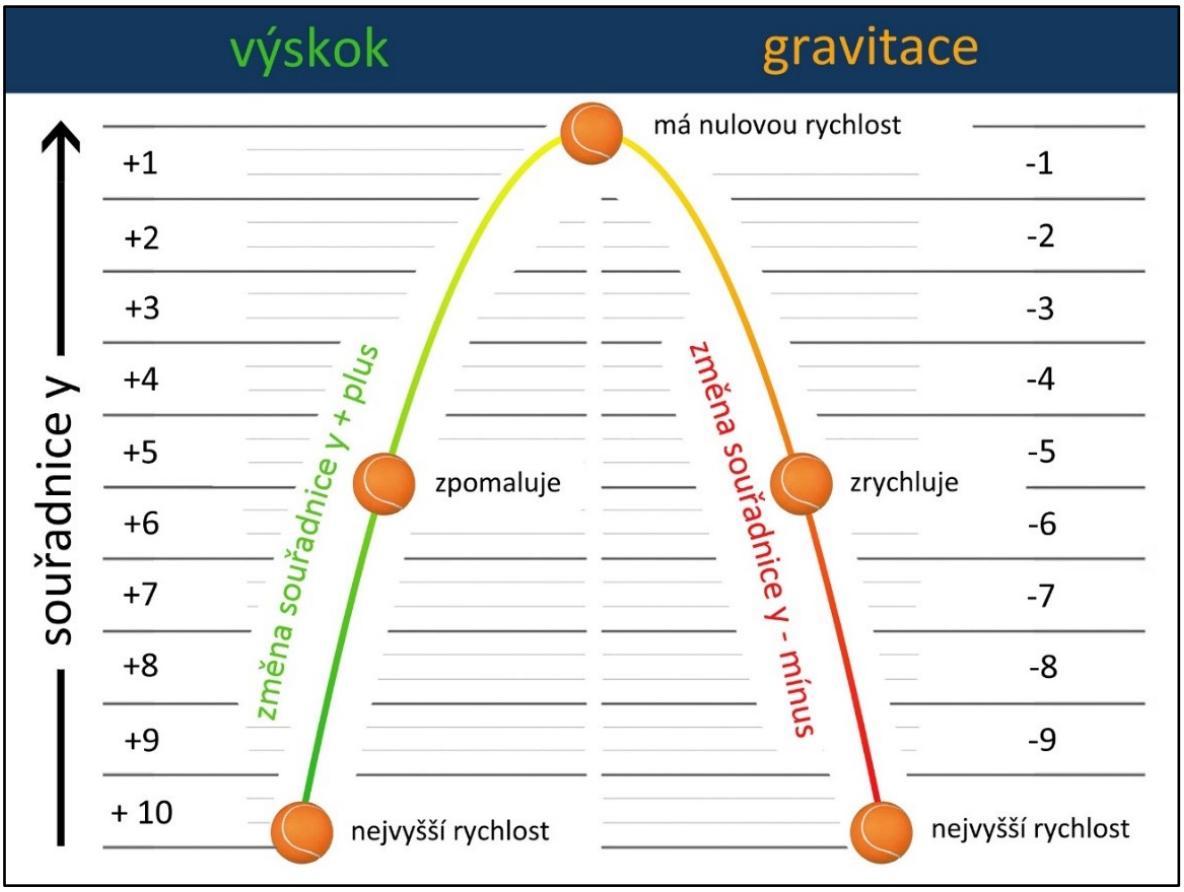


Obrázek 24 – Příklad algoritmu s konceptem Proměnná (proměnná „Body“)

Obrázek 25 – Příklady využití konceptu Proměnná ve Scratch (proměnné „mezičas“ a „Hop“)



Obrázek 26 – Příklad kódu s konceptem Proměnná ve Scratch (proměnné „rychlost skoku“ a „Body“)



Obrázek 27 – Pomůcka pro vysvětlení pojmu rekurze

Poznámky pro učitele

Protože pojmy zrychlení a zpomalení jsou pro žáky 5. a 6. ročníku ještě neznámé, je vhodné jim plynulost skoku vysvětlit na konkrétním příkladu. Pro žáky není obtížné určit, že předmět vyhozený do vzduchu:

1. má nejvyšší rychlost na počátku letu
2. se v nejvyšším bodě zastaví, tedy že jeho rychlost je nulová

Z toho lze při diskusi s žáky odvodit, že při stoupání předmět zpomaluje a naopak při klesání zrychluje.

Pro naprogramování skoku je tedy vhodné použít postupnou změnu souřadnice Y. Po výuce proměnné lze sekvenci postupného přičítání a odečítání souřadnice Y přepracovat na rekurzivní cyklus.

Ukázka výsledného souboru na https://scratch.mit.edu/projects/228629577/.

4.3.11 Lekce 11 – Klonování

Programování ve Scratch umožňuje vytváření kopií postav neboli klonů. Všechny namnožené postavy mají shodné vlastnosti jako originál a řídí se stejným scriptem.

Tabulka 17 – Příkazy konceptu Klonování ve Scratch

Příkazy klonuj sebe a klonuj jinou postavu	
Událost – když startuje můj klon	
Příkaz zruš tento klon	

Cíl výuky

- Žák dokáže vysvětlit pojem klonování z programu Scratch.
- Vhodně používá příkazy konceptu klonování.

Postup práce

(1) Učitel představí žákům příkazy „Klonuj sebe“ – „Klonuj sprite“ – „Když startuje můj klon“ a „Zruš tento klon“.
(2) Žáci otevřou webový prohlížeč – Google Chrome, Microsoft Edge, Mozzilla Firefox apod.
(3) Zadají adresu webu Scratch (https://Scratch.mit.edu), záložka „Tvořit“.
(4) Žáci načtou vzorový soubor Klonování-zadání. (https://scratch.mit.edu/projects/228538846/)
(5) Úkolem žáků je přidat do projektu padající sněhové vločky pomocí konceptu klonování. Rychlejší žáci mohou po startu klonu naprogramovat jeho pohyb libovolným směrem a zrušení klonu až v případě dotyku s okrajem scény.
(6) Prezentace výsledných programů, vzájemné hodnocení.

Poznámky pro učitele

Originální postavu, kterou budeme klonovat, je vhodné skrýt, aby nebyla vidět. Každý klon může měnit své vlastnosti dle potřeby – velikost, barvu, umístění v souřadnicovém systému apod.

Pro vytvoření dojmu hloubky prostoru je vhodné větší vločky umístit do popředí, před postavu robota a naopak malé vločky do pozadí, za postavu robota.

(viz https://scratch.mit.edu/projects/228553832/)

Nejrychlejší žáci mohou projekt přetvořit na hru, kdy úkolem hráče bude zamezit, aby vločky dopadaly na robota. V tomto případě je vhodné zmenšit postavu robota a k vločkám přidat script, který způsobí zmizení vločky po kliknutí na ni či po dotyku vločky s jinou postavou (spritem)

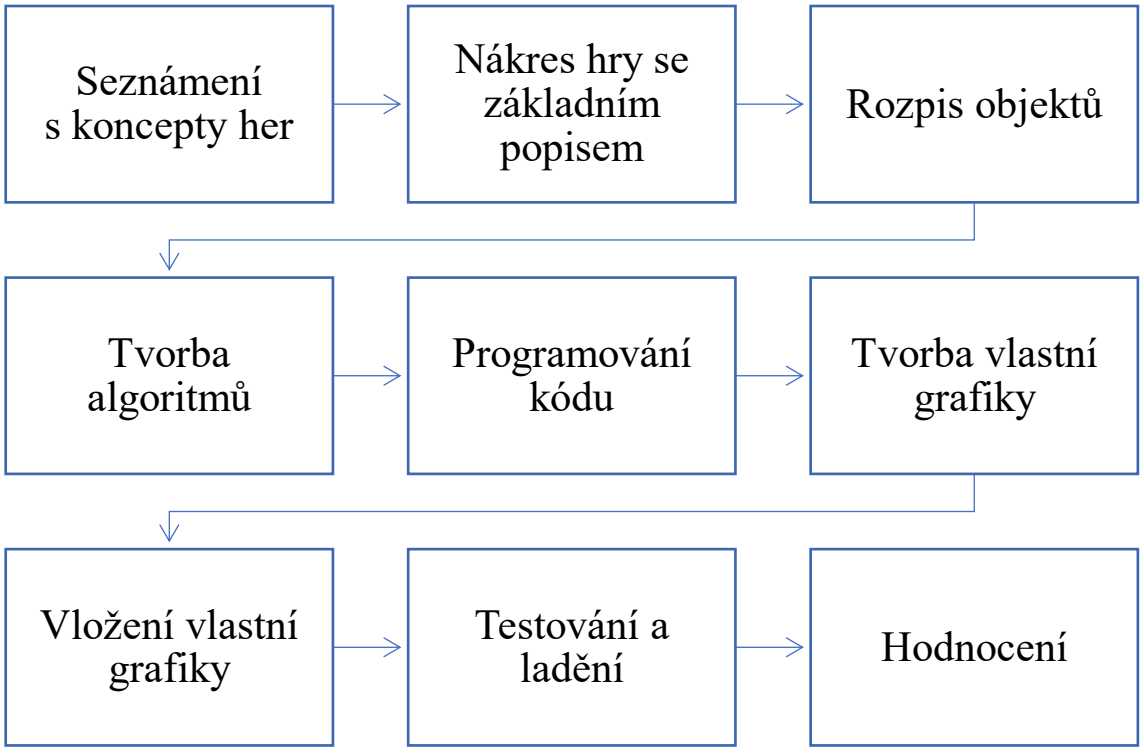
(viz https://scratch.mit.edu/projects/228571000/)

Obrázek 28 – Ukázky úlohy zaměřené na klonování

Obrázek 29 – Příklad vláken z úlohy Klonování

4.3.12 Závěrečný projekt

Po výuce konceptů programování byl na její konec zařazen blok hodin, v jejichž průběhu žáci pracovali na závěrečném projektu, kterým byla tvorba vlastní hry v prostředí Scratch. Projekt byl rozdělen do několika částí. V první fázi byli žáci seznámeni s koncepty několika her, u nichž byl předpoklad, že obtížnost jejich tvorby odpovídá šíři osvojených dovedností všech žáků. Důvodem byla podpora motivace žáků a především snaha eliminovat nepřiměřené představy žáků o možnostech výsledných her. V druhé fázi žáci vytvořili nákres hry se základním popisem a specifikovali jednotlivé objekty z hlediska vlastností a interakcí mezi sebou. Následovalo programování kódu. V rámci mezipředmětových vztahů si žáci v hodinách výtvarné výchovy připravili vlastní grafiku pro pozadí a objekty ve hře. Na závěr proběhlo testování, ladění a hodnocení jednotlivých her.

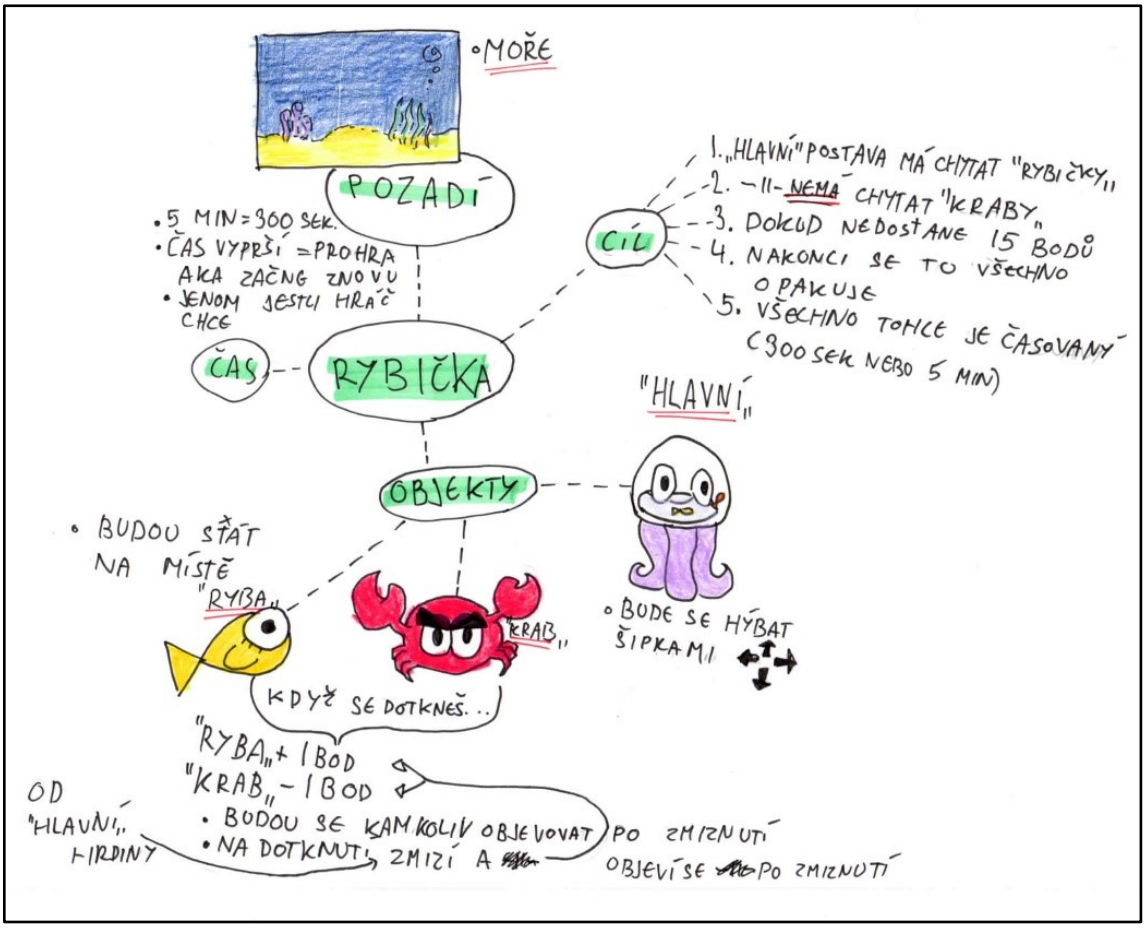


Obrázek 30 – Postup tvorby závěrečného projektu

Žáci měli možnost zvolit, zda budou pracovat individuálně či ve dvojicích. Přestože v průběhu výuky volili žáci až na výjimky možnost pracovat ve dvojicích, většina z nich se

rozhodla pracovat samostatně. Jako důvod žáci uvedli, že chtějí na projektech pracovat také mimo vyučování z domova a každý si chce vytvořit hru vlastní.

S ohledem na rozsah projektu byla část příprav realizována v rámci mezipředmětových vztahů. V průběhu dvou hodin výtvarné výchovy nakreslili žáci obrázek hry a do obrázku začlenili popis hry a základní interakce jednotlivých objektů. Tato příprava pomohla žákům ujasnit si a rozvrhnout další činnosti, například tvorbu grafiky pro jednotlivé objekty (viz Obrázek 31).



Obrázek 31 – Návrh hry

V rámci předmětu český jazyk vyplnili žáci pracovní list (viz Obrázek 36), který obsahoval pole pro jednoduché znázornění jednotlivých objektů a řádky pro jejich popis. Úkolem žáků bylo vlastními slovy popsat, jak budou jednotlivé postavy ve hře ovládány hráčem, jak budou navzájem interagovat a jaké podmínky pro ně budou platit. Programování her již probíhalo pouze v hodinách informatiky.

4.4 Analýza dopadu výuky

Pro zkoumání dopadu výuky základů programování ve Scratch bylo využito především metod soustavného pozorování v průběhu výuky, analýzy závěrečných projektů žáků, test porozumění konceptům Scratch a závěrečné dotazníkové šetření.

4.4.1 Analýza výuky

Výuka základů programování ve Scratch byla rozvržena do tří vyučovacích bloků, které obsahovaly celkem 11 výukových lekcí a tvorbu závěrečného projektu. Osmi lekcím byl vyhrazen čas vždy v rozsahu jedné vyučovací hodiny (45 min.), třem lekcím pak vždy dvě vyučovací hodiny. Závěrečný projekt byl realizován v šesti vyučovacích hodinách informatiky, čtyřech vyučovacích hodinách výtvarné výchovy a jedné hodině českého jazyka.

V prvním vyučovacím bloku, který trval čtyři vyučovací hodiny, byli žáci seznámeni s pojmem algoritmus, s prostředím programu Scratch a s možnostmi nastavení vlastností objektů v Scratch. V druhém vyučovacím bloku zaměřeném na konkrétní programovací prvky žáci absolvovali lekce objasňující sekvence, cykly, paralelizaci, události, zprávy, podmínky a operátory, proměnné a klonování. V úvodu každé lekce žáci společně, za podpory vyučujícího, tvořili jednoduchý algoritmus, založený na principu programovacího konceptu probíraného v příslušné lekci. Dále byli žáci seznámeni s příslušnými programovacími koncepty a odpovídajícími příkazy v programu Scratch. V následující části žáci individuálně řešili úlohu založenou na probraném konceptu. Na závěr lekce proběhla společná kontrola výsledků a porovnání použitých kódů. Žákům, kteří nebyli schopni samostatně programy tvořit, byla průběžně poskytována podpora a pomoc ze strany vyučujícího a rychlejších spolužáků. S ohledem na problémy, které vyvstaly v průběhu výuky, trval tento blok celkem 14 vyučovacích hodin.

S procedurami, testováním a laděním programů se žáci seznámili v třetím vyučovacím bloku, během tvorby závěrečných projektů, kterým bylo věnováno šest vyučovacích hodin.

Tabulka 18 – Struktura výuky

Výukový blok	Téma	Číslo lekce	Počet hodin	Datum
Algoritmizace a programování	Algoritmus	Lekce č. 1	1	16. 10. 2017–6. 11. 2018
	Seznámení s programem Scratch	Lekce č. 2	1	
	Vlastnosti objektů	Lekce č. 3	1	
	Sekvence	Lekce č. 4	1	
Výuka konceptů programování	Cyklus	Lekce č. 5	1	6. 1. 2017–12. 2. 2018
	Paralelizace	Lekce č. 6	1	
	Události	Lekce č. 7	1	
	Zprávy	Lekce č. 8	2	
	Když (If, operátory)	Lekce č. 9	2	
	Proměnná	Lekce č. 10	1	
	Klonování	Lekce č. 11	2	
Závěrečný projekt	Programování		6	12. 2. 2018–20. 4. 2018
	Testování			
	Ladění			
	Procedury			
Celkem			20	16. 10. 2017–20. 4. 2018

Tabulka 18 znázorňuje členění jednotlivých vyučovacích jednotek, ve kterých probíhala výuka základů programování. Metodický přístup kombinoval aktivity typu „CS unplugged“ bez počítače s aktivitami na počítači v prostředí Scratch.

Tabulka 19 – Struktura vyučovacích hodin

Fáze vyučovací hodiny	Činnost	Aktivity
evaluace osvojených vědomostí a dovedností žáků	aktivity pro upevnění osvojených znalostí a dovedností (společná tvorba algoritmu s programovacím konceptem probíraným v předchozí lekci, aktivity s kartami zpráv, paralelní vykonávání zadaných činností…)	bez počítače
sdělení tématu a evokace souvisejících, již osvojených, vědomostí a dovedností z předchozí výuky	výklad učitele rozhovor se žáky (otázky a odpovědi)	bez počítače
prezentace nových poznatků	výklad učitele rozhovor se žáky (otázky a odpovědi)	bez počítače
	prezentace ukázkových programů (bez zobrazení kódu)	s počítačem
aplikace nových poznatků	společné sestavení algoritmu s probíraným programovacím konceptem	bez počítače
	vlastní objevování žáků řešení zadaných úloh konstrukce kódu ve Scratch	s počítačem
ověřování, zda žáci porozuměli probíranému tématu a osvojili si požadované dovednosti a vědomosti	prezentace jednotlivých řešení	s počítačem
	diskuse nad vybranými řešeními, volba nejvhodnějších řešení rozhovor se žáky (otázky učitele a odpovědi žáků)	bez počítače
závěr	shrnutí tématu, dotazy žáků, zadávání dobrovolných domácích úkolů	bez počítače

Vzhledem k faktu, že výuka probíhala ve čtyřech skupinách A, B, C a D, byly i výsledky jednotlivých testů vyhodnocovány s ohledem na rozmístění žáků v těchto skupinách. K ověřování dopadu výuky byly použity metody pozorování, rozhovorů s žáky, testové úlohy zaměřené na algoritmizaci (TEST A2) a testové úlohy zjišťující schopnost žáků porozumět kódům a algoritmům (TEST B1), se kterými se žáci setkali v průběhu výuky.

Seznámení s programem

Rozložení pracovní plochy pro práci se Scratch se ukázalo být poměrně přehledné a uživatelsky přívětivé. Přesto některým žákům její členění činilo menší obtíže. Ikony pro často využívané činnosti kopírování, změnu velikosti objektů i funkce aktivní nápovědy jsou umístěny v prostoru, kde lze předpokládat hlavní ovládací prvky, ale jsou malé a málo výrazné. Navíc se nacházejí v těsné blízkosti barevných ikon pro spuštění a zastavení programu a také kategorií programovacích příkazů, které výrazně upoutávají pozornost

žáků. Někteří žáci rovněž opakovaně hledali vlastnosti objektů, ukryté pod malou ikonou „i“ v levém horním rohu náhledu postavy, která se však zobrazuje pouze u vybraného objektu. Pro žáky 5. ročníku bylo zpočátku výuky rovněž matoucí přepínání objektů z bitmapového do vektorového režimu a naopak. S příslušnými grafickými režimy nebyli doposud seznámeni, navíc má každý režim panel nástrojů umístěn odlišně na opačných stranách kreslicí plochy. Přes tyto nedostatky nebylo samotné prostředí programu pro žáky překážkou při výuce ani při tvorbě závěrečných projektů.

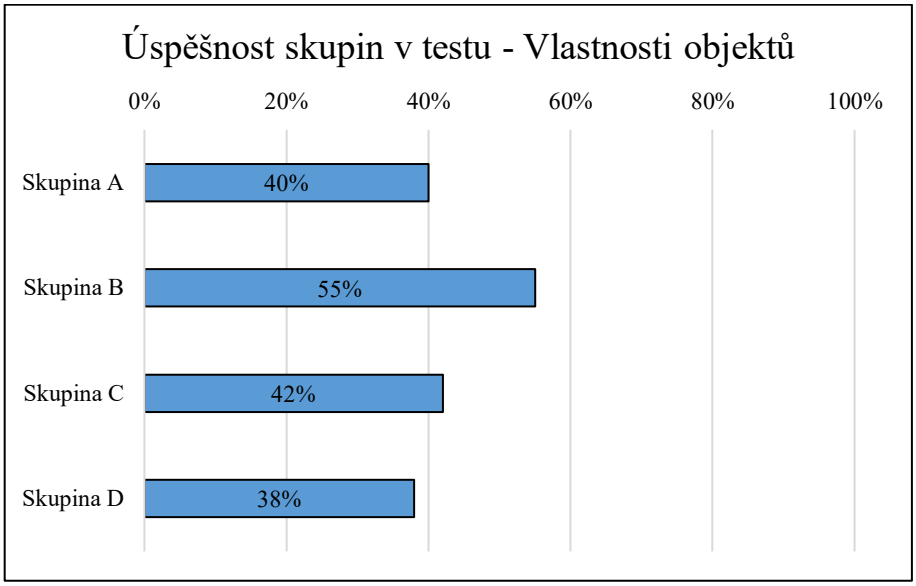
Vlastnosti objektů

Vybrané vlastnosti objektů činily některým žákům v průběhu celého pedagogického experimentu obtíže. Jednalo se především o pozici objektu v souřadnicovém systému, který používá program Scratch. Se zápornými čísly se žáci skupin zařazených do experimentu doposud nesetkávali. Cvičení aritmetického typu, ve kterých žáci pracují s číselnou osou a teploměrem, jsou v rámci školního vzdělávacího programu zařazeny až do 6. ročníku a jsou vyučovány v předmětu matematika v průběhu druhého pololetí.

Na dotaz, kde se nachází souřadnice X=0 a Y=0, sice většina žáků dokázala po lekci odpovědět, že uprostřed scény, nicméně při programování pohybu postav pomocí změny souřadnic měla přibližně třetina žáků skupiny A a dvě třetiny žáků skupiny B obtíže s určováním správného směru u záporných hodnot a opakovaně se dopouštěla chyb. Ve skupinách C a D se obdobný problém týkal vždy dvou žáků.

Druhým problémem se ukázala neznalost rozdělení kruhu na úhly a stupně. Protože však v prostředí Scratch u nastavení vlastností existuje kurzorem nastavitelná volba směru s ukazatelem zvoleného směru objektu, využívali žáci tuto možnost bez bližšího pochopení tohoto geometrického pojmu.

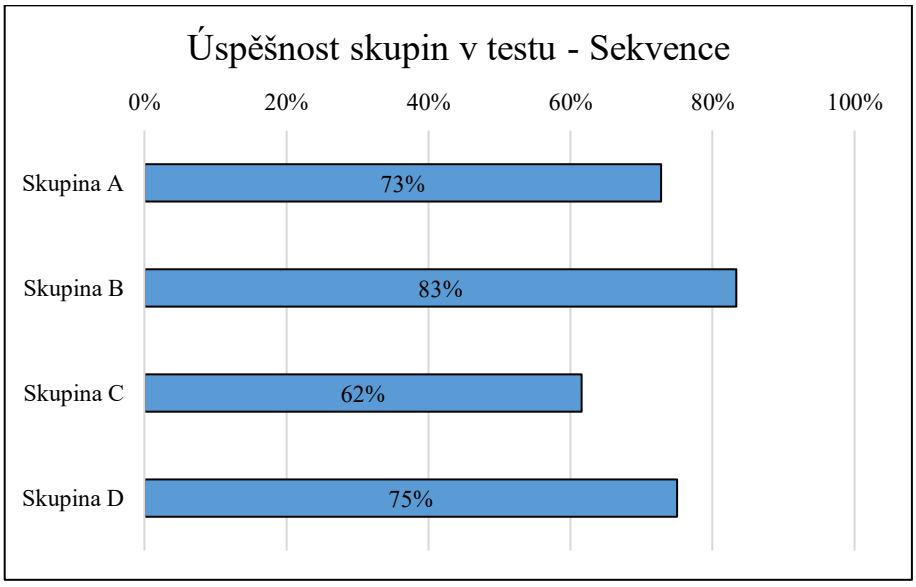
V testu (TEST B1, viz Příloha C) ověřujícím dopady výuky se vlastnostem objektu ve Scratch věnovala otázka č. 7, ve které měli žáci za úkol vybrat z kódů ten, který po startu posune objekt ze středu scény doleva dolů, a otázka č. 16, ve které měli žáci napsat pět nastavitelných vlastností objektu ve Scratch. Nízká úspěšnost správných odpovědí u této otázky může být dána faktem, že se jednalo o otázku s otevřenou odpovědí.



Graf 13 – Úspěšnost skupin v testu – Vlastnosti objektů

Sekvence

S pochopením konceptu sekvence neměli žáci při samotné výuce problém. Velmi brzy si uvědomili, že aby program řádně „fungoval“, je třeba dodržet správnou posloupnost příkazů. U jednoduchých programů, které žáci vytvářeli, se problém s chybným pořadím příkazů vyskytoval spíše ojediněle. Porozumění konceptu Sekvence se v testu (TEST B1, viz Příloha C) zabývala otázka č. 3. Výsledky testu však vypovídají, že řádné porozumění všem příkazům v sekvenci není zcela samozřejmé.

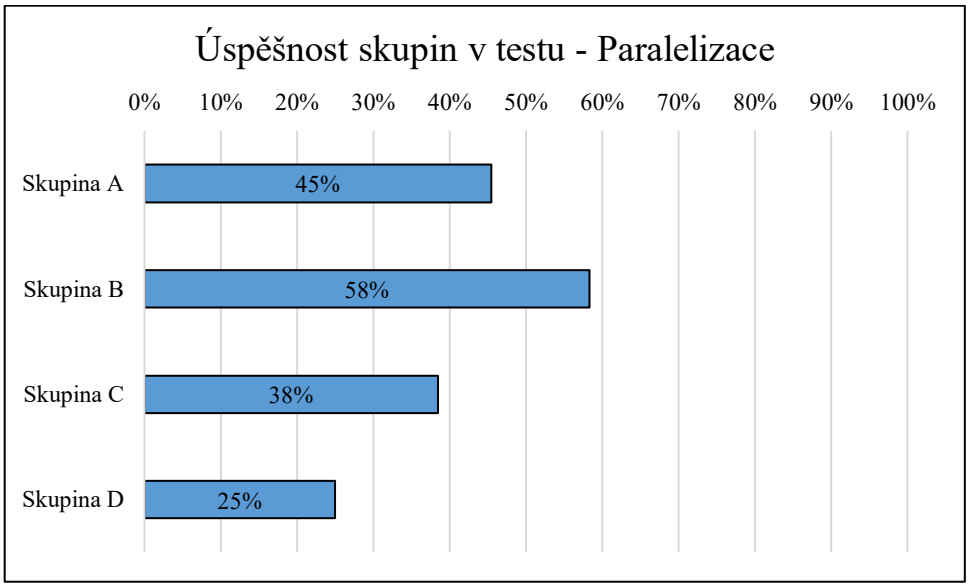


Graf 14 – Úspěšnost v testu – Sekvence

Paralelizace

Termín paralelní je pro žáky ve věku 10–12 let většinou neznámý. Spojují si jej nejčastěji ve významu slovního spojení paralelní světy, aniž by významům těchto termínů rozuměli. Pro snazší porozumění žáků principu paralelizace se osvědčilo tento termín přiblížit názorně. Jako vhodné se ukázalo přehrání videa s paralelním slalomem snowboardingu či lyžování. Účinné rovněž bylo zařazení hry, při níž měli vždy dva žáci za úkol vykonávat současně zadané činnosti se zavřenýma očima, podle tleskání v rytmu třetí osoby. Děti mohou rovněž tančit ve dvojicích, kdy každý z partnerů má určeny jiné kroky.

Míru porozumění konceptu paralelizace ověřovala otázka č. 8 (TEST B1, viz Příloha C), v níž měli žáci za úkol zvolit dva ze čtyř kódů, které začnou a skončí ve stejný okamžik.



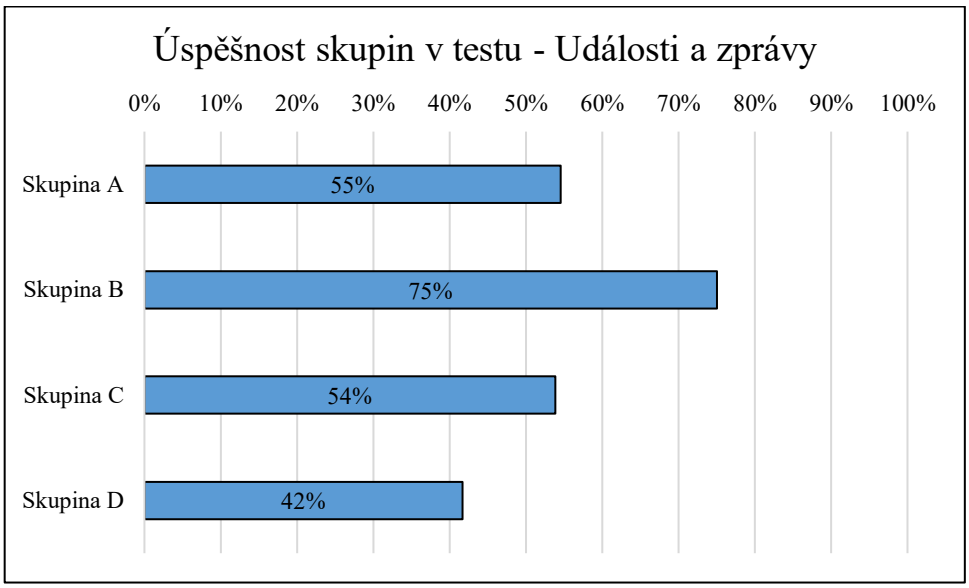
Graf 15 – Úspěšnost v testu – Paralelizace

Události a Zprávy

Samotná výuka proběhla bez komplikací. V případě výuky konceptu zpráv se však brzy ukázalo, že překlad některých příkazů z anglického do českého jazyka může být pro mladší žáky matoucí. Přestože po úvodním výkladu při samostatném řešení úlohy zaměřené na synchronizaci zobrazování textů „hovořících“ postav postupovali žáci správně, velmi brzy začali zaměňovat příkaz „Rozešli zprávu“, za příkaz „Říkej (text)“. Přibližně polovina žáků skupiny A a téměř všichni žáci skupiny B i po opakovaném výkladu a rozboru těchto příkazů postupovali chybně. Ve skupinách C a D (žáků z 6. třídy) se tento problém vyskytoval

zřídka. Přesto byl ve všech skupinách společně s žáky proveden rozbor těchto příkazů v rámci aktivit bez počítače, při kterém byla hledána slova přesněji vystihující funkci těchto příkazů. Výsledkem byla shoda na slovních spojeních „Rozešli signál (pokyn)“ v případě události „Rozešli zprávu“ a „Zobraz bublinu s textem“ namísto příkazu „Říkej“. Tato „synonyma“ byla napsána na tabuli po další tři vyučovací hodiny, dokud si žáci neutvořili vazby mezi těmito spojeními a nepochopili pravý význam těchto příkazů. Současně byla na počátek další tří lekcí zařazena aktivita s vytištěnými kartami příkazů „Rozešli zprávu“ a „Po přijetí zprávy“ (viz Lekce 8 – Zprávy, Obrázek 21).

Porozumění konceptu Událostí a Zpráv se věnovala otázka č. 12 (TEST B1, viz Příloha C).



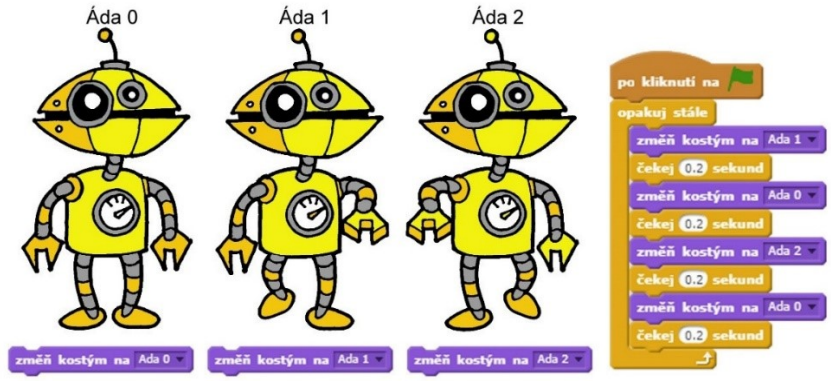
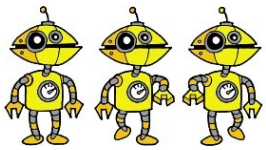
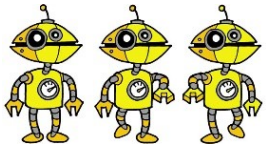
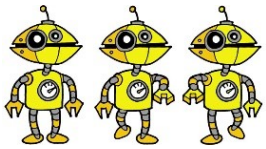
Graf 16 – Úspěšnost v testu – Události a zprávy

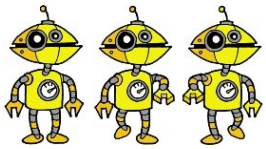
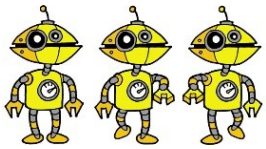
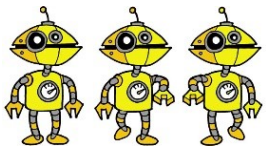
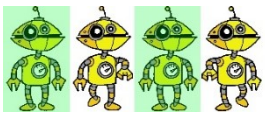
Cyklus

S pojmem cyklus žáci spojují jako synonymum slovo opakování. Se základním pochopením cyklu, tedy části kódu, která se opakuje, neměli žáci větší problém. Téměř všichni však při prvních pokusech u samostatného úkolu animovat objekt robota pomocí změny jeho tří kostýmů použili uvnitř cyklu pouze příkaz Další kostým, v kombinaci s příkazem „Čekej 0.1 sekund“, nebo chybně řadili sekvenci příkazů.

Po diskusi, zda neexistuje jiný způsob, který by umožnil použít pouze kombinaci příkazů „Další kostým“ a „Čekej 0.1 sekund“, obdrželi žáci čas na hledání řešení. Tímto řešením je opětovné vložení kostýmu č. 0 (obě nohy robota na zemi) a seřazení všech čtyř kostýmů

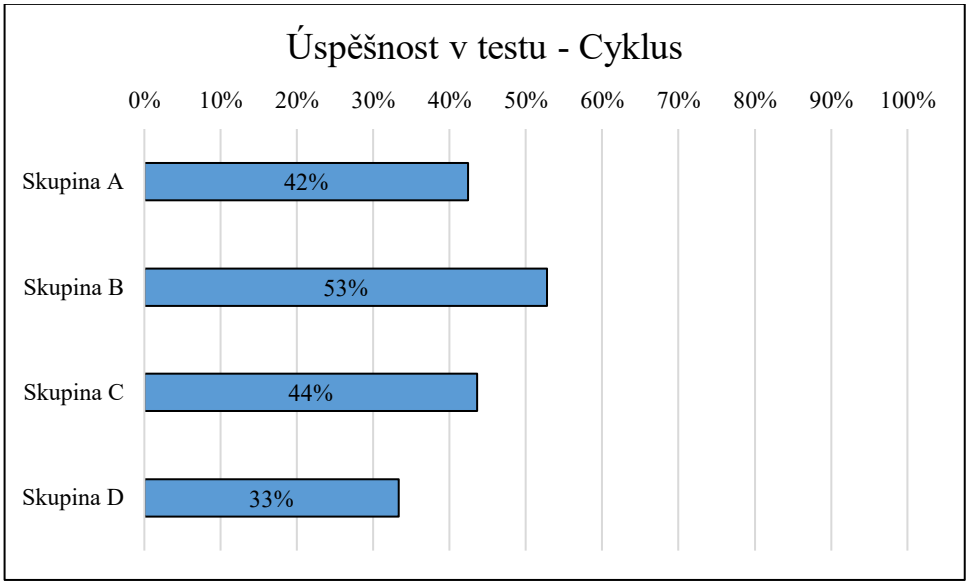
v pořadí 1-0-2-0. Tím současně dosáhneme toho, že při skončení animace bude robot stát oběma nohama na zemi. Toto řešení však přes doplňující návodné otázky žádný z žáků nenalezl.

<table>
<tr><td colspan="3"> 
</td></tr>
<tr><td>  </td><td> 
</td><td>Chybné řešení.
Zvolili jako první řešení téměř všichni žáci.
Dochází ke střídání kostýmů v pořadí „kostým 0 → kostým 1 → kostým 2“.
Existuje 66% pravděpodobnost, že po skončení programu zůstane robot „s jednou zdvihnutou nohou“.</td></tr>
<tr><td>  </td><td> 
</td><td>Chybné řešení.
Většina žáků použila jako druhé řešení.
Dochází ke střídání kostýmů v pořadí „kostým 0 → kostým 1 → kostým 2“.
Z kostýmu 2 na kostým 1 dojde ke změně ihned => v animaci nelze zrakem zaznamenat kostým 2.
Existuje 50% pravděpodobnost, že po skončení programu zůstane robot „s jednou zdvihnutou nohou“.</td></tr>
<tr><td>  </td><td> 
</td><td>Chybné řešení.
Většina žáků použila jako opravu druhého řešení.
Dochází ke střídání kostýmů v pořadí „kostým 0 → kostým 1 → kostým 2“.
Existuje 66% pravděpodobnost, že po skončení programu zůstane robot „s jednou zdvihnutou nohou“.</td></tr>
</table>

		Chybné řešení. Třetina žáků použila jako opravu druhého řešení. Dochází ke střídání kostýmů v pořadí „kostým 1 → kostým 2“. Po skončení programu zůstane robot vždy „s jednou zdvihnutou nohou“.
		Správné řešení. Polovina žáků použila jako opravu druhého řešení. Dochází ke střídání kostýmů v pořadí „kostým 0 → kostým 1 → kostým 0 → kostým 2“. Existuje však 50% pravděpodobnost, že po skončení programu zůstane robot „s jednou zdvihnutou nohou“.
		Správné řešení. Pět žáků skupin C a D použilo jako první řešení. Ostatní použili díky dopomoci jako poslední řešení. Dochází ke střídání kostýmů v pořadí „kostým 0 → kostým 1 → kostým 0 → kostým 2“. Po skončení programu zůstane robot „s oběma nohama na zemi“.
		Správné řešení. Přes značnou dopomoc ve formě návodných otázek žádný z žáků všech skupin nenašel řešení ve formě doplnění řady kostýmů o kopii kostýmu 0. Dochází ke střídání kostýmů v pořadí „kostým 1 → kostým 0 → kostým 2 → kostým 0“.

Obrázek 32 – Možnosti řešení vytvoření dojmu chůze pomocí cyklu

Porozumění konceptu cyklu ověřovaly v testu (TEST B1, viz Příloha C) otázky č. 2, 4 a 5.



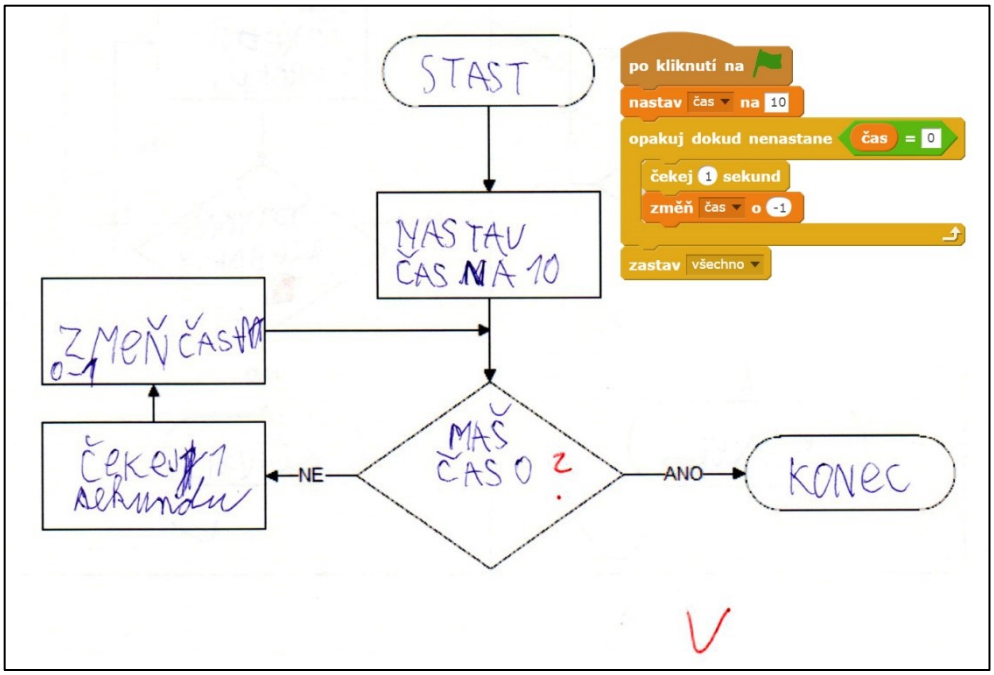
Graf 17 – Úspěšnost v testu – Cyklus

If, operátory

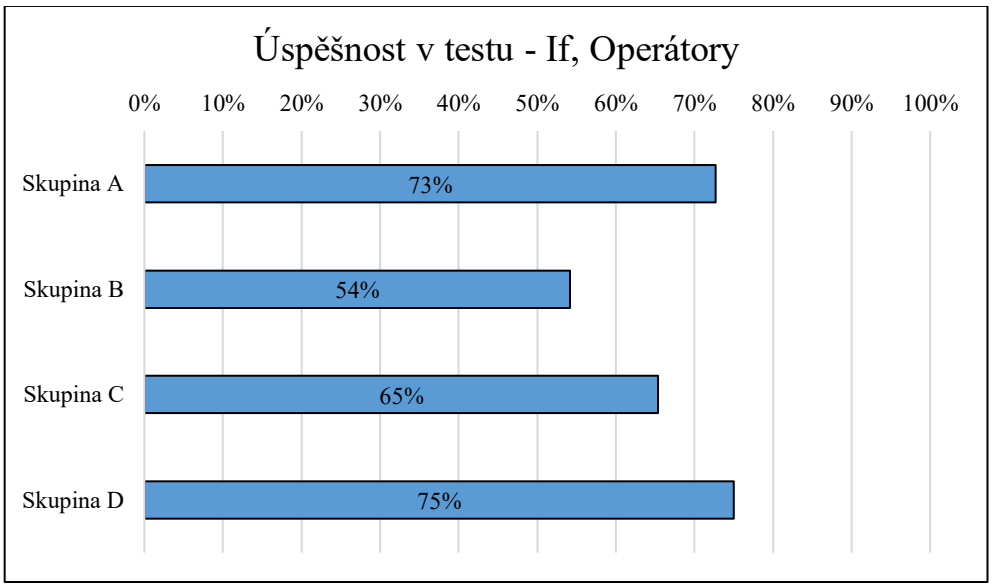
S porozuměním podmíněným příkazům neměli žáci žádné skupiny vážnější problém. Velmi brzy pochopili, že se jedná o otázku, na kterou lze odpovědět pouze ANO nebo NE.

Osvojení konceptu podmíněného příkazu napomohlo, že vytváření otázek bylo průběžně procvičováno pomocí tvorby algoritmů v úvodu většiny lekcí.

V testu porozumění programovacím konceptům (TEST B1, viz Příloha C) žáci dosahovali při tvorbě podmíněných příkazů velmi dobrých výsledků. Nejčastější chybou, které se žáci dopouštěli, byla absence otazníku na konci otázek. Za toto pochybení však nebyly žákům strhávány body, neboť žáci ve většině případů prokázali porozumění principu podmíněných příkazů (viz Obrázek 33).



Obrázek 33 – Ukázka úlohy Test B1 – Operátory

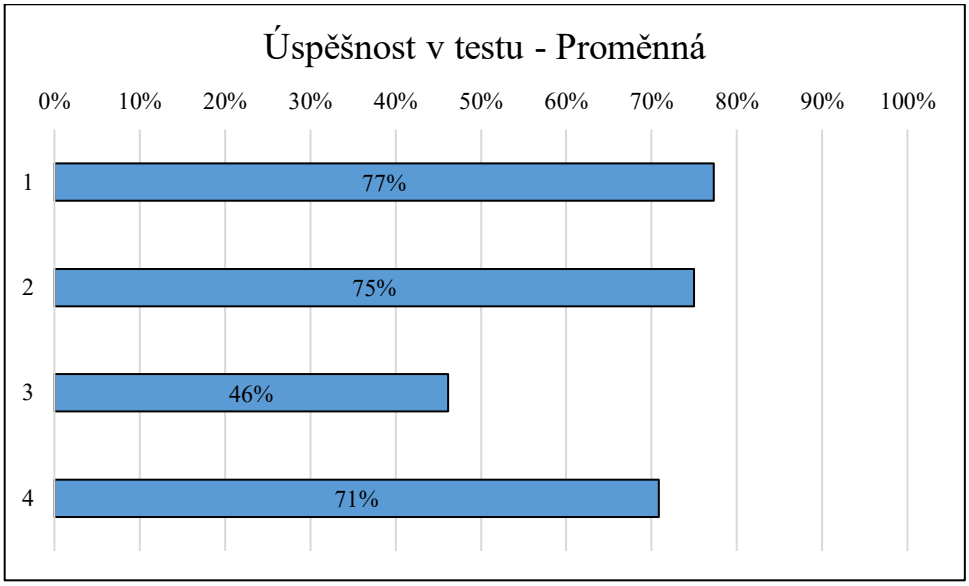


Graf 18 – Úspěšnost v testu – If, Operátory

Proměnná

Přestože Proměnná je pro žáky abstraktním pojmem, výsledky testu ukazují, že konceptu proměnné žáci porozuměli dobře. Tomuto konceptu se v testu (TEST B1, viz Příloha C) věnovaly otázky č. 10 a 11. Úkolem žáků bylo určit, jakou hodnotu budou mít proměnné

po skončení konkrétních scriptů? S výjimkou skupiny C, která v testu u otázek zaměřených na koncept paralelizace dosáhla pouhých 46 %, všechny ostatní skupiny překročily hranici 71 %, skupina A dokonce 77 %.



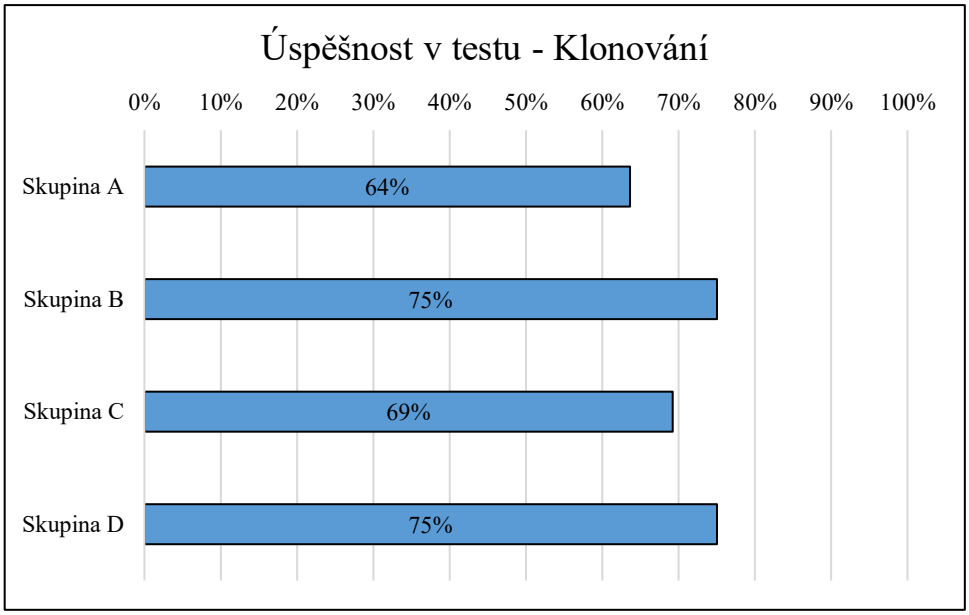
Graf 19 – Úspěšnost v testu – Proměnná

Klonování

Porozumění principu a tvorba kódů s konceptem klonování nečinila žákům potíže. Problém představovalo jejich funkční začlenění do kontextu celých programů. K tomu bylo zapotřebí tyto příkazy kombinovat s příkazy určujícími vlastnosti postav. Jednalo se především o příkazy reagující na start nového klonu, například zobrazení postavy, její umístění na náhodnou pozici, určení časové prodlevy mezi startem klonů apod.

Porozumění konceptu klonování ověřovala v testu (TEST B1, viz Příloha C) otázka č. 6**.** Žáci měli za úkol vybrat ze čtyř kódů ten, který po obdržení zprávy start objekt nejprve skryje a každých 10 sekund vytvoří jeho klon. Úspěšnost správných odpovědí u všech žáků

přesáhla v průměru 70 %, což řadí v porozumění tento koncept na třetí místo za algoritmizaci a sekvence.

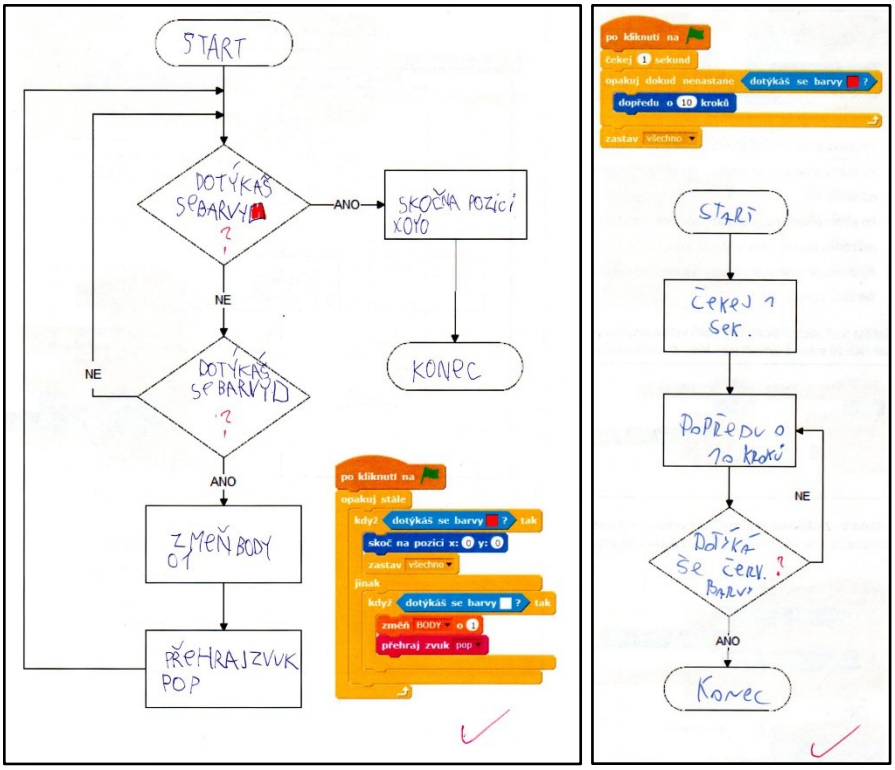


Graf 20 – Úspěšnost v testu – Klonování

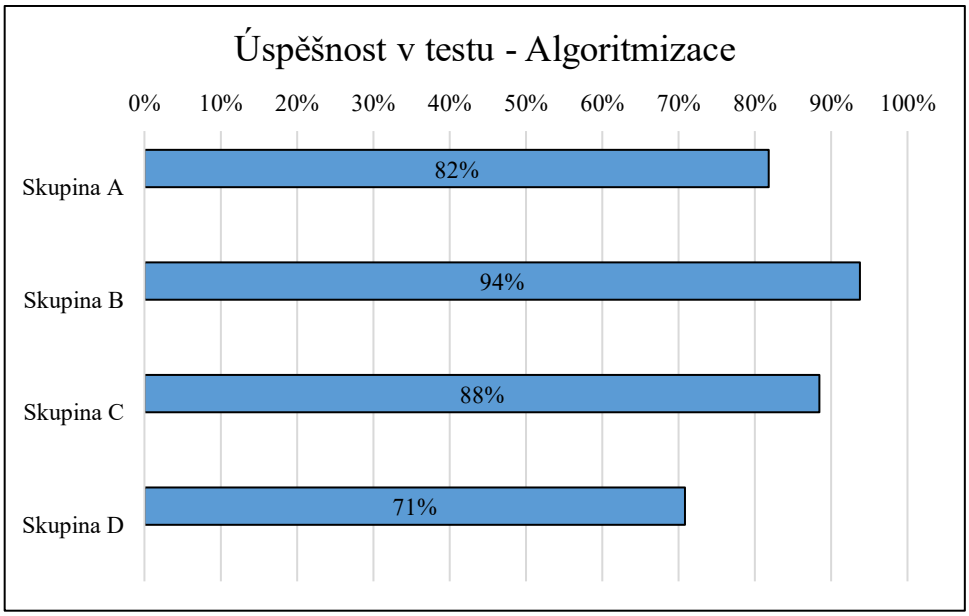
Algoritmizace

S pojmem algoritmus, s jeho vlastnostmi a druhy zápisu algoritmů pomocí vývojových diagramů, byli žáci seznámeni v úvodní lekci výuky. Součástí tohoto tématu byl pracovní list (Příloha B), v němž měli žáci navrhnout postup práce či návod, který bude jednoznačný, opakovatelný, použitelný ve více situacích a konečný.

S ohledem na cíle práce bylo v testu (TEST B1, viz Příloha C) ověřujícím porozumění jednotlivým konceptům programování věnováno celkem pět otázek zaměřených na schopnost porozumět algoritmům. Úkolem žáků bylo doplnit do vývojových diagramů příkazy a podmínky tak, aby odpovídaly příslušným scriptům v Scratch.



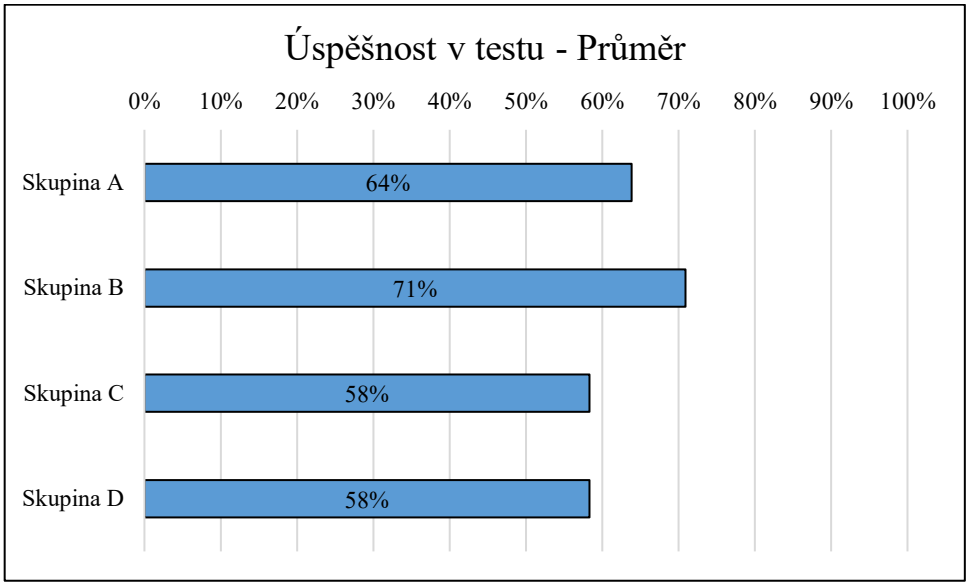
Obrázek 34 – Ukázka úlohy TEST B1 – Algoritmizace



Graf 21 – Úspěšnost v testu – Algoritmizace

V případě porozumění konceptu Algoritmizace žáci všech skupin dosáhli v testu (TEST B1, viz Příloha C) nejvyšší úspěšnosti. Oproti ostatním konceptům, které byly probírány

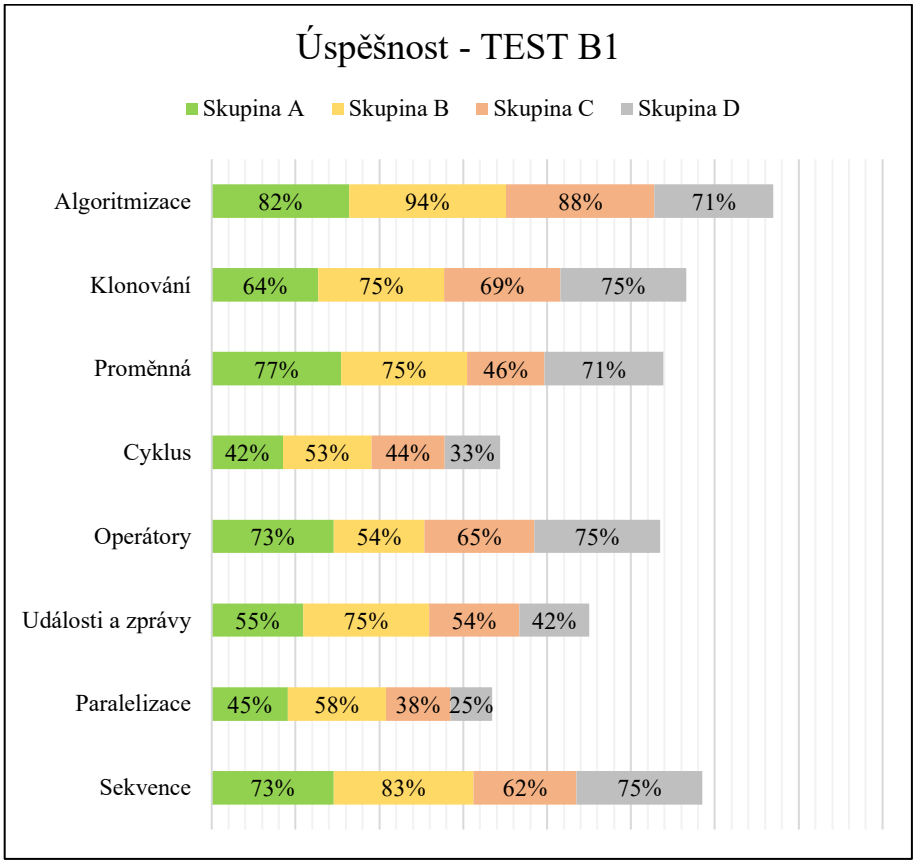
v jednotlivých lekcích, byl v úvodu každé lekce v rámci aktivit bez počítače sestavován algoritmus obsahující příslušný probíraný koncept. Lze tedy předpokládat, že vyšší míra osvojení znalostí a dovedností v oblasti porozumění algoritmům a jejich sestavování, oproti osvojování ostatních programovacích konceptů, byla způsobena zaměřením pedagogického experimentu.



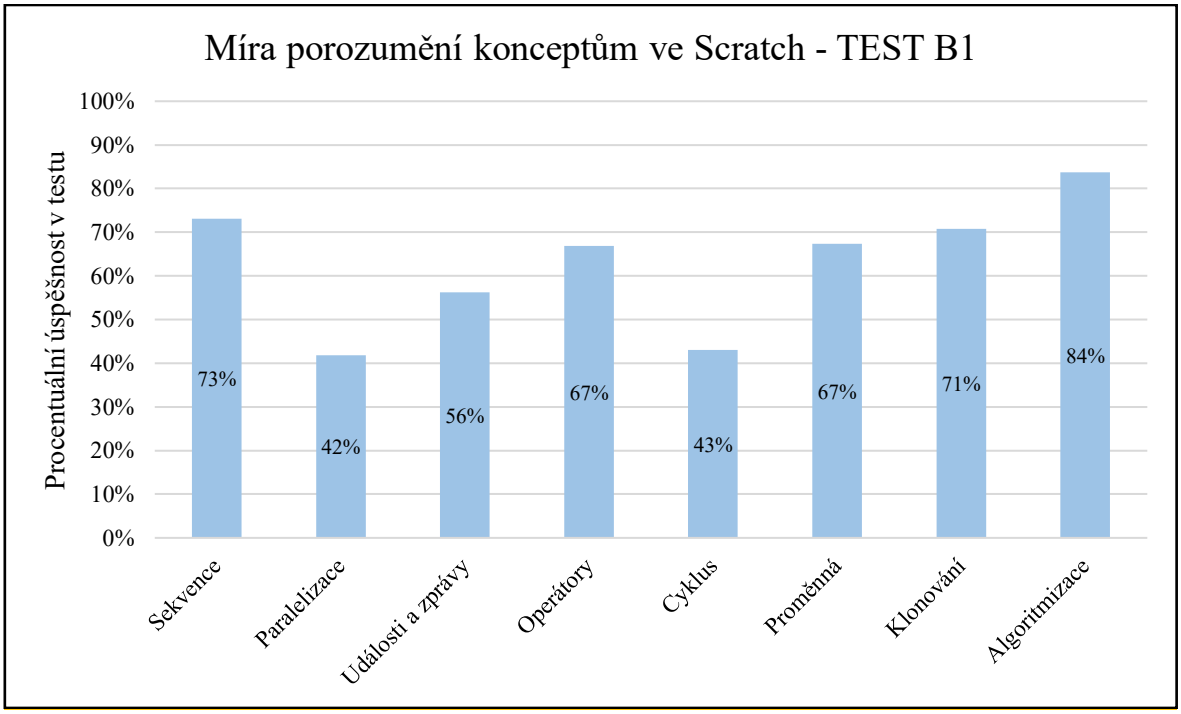
Graf 22 – Úspěšnost skupin – TEST B1

Výsledky testu (TEST B1, viz Příloha C) ukazují, že jednotlivé programovací koncepty si většina žáků osvojila. Je rovněž zřejmé, že po ukončení výuky základů programování si žáci nejlépe osvojili znalosti a dovednosti z oblasti Algoritmizace. Nejhorších výsledků dosahovali u konceptů Cyklus a Paralelizace (viz Graf 24).

Nejlepších výsledků v testu ověřujícím porozumění probíraným programovacím konceptům dosáhli žáci skupiny B, jejichž procentuální úspěšnost činila celých 71 %. Druhého nejlepšího výsledku dosáhli žáci skupiny A s 64 % správných odpovědí. Skupiny C a D odpověděly shodně na 58 % otázek. Lépe si tedy vedli žáci 5. třídy, kteří předčili své starší spolužáky z 6. třídy v průměru o 9,5 %.



Graf 23 – Procentuální úspěšnost skupin v TEST B1



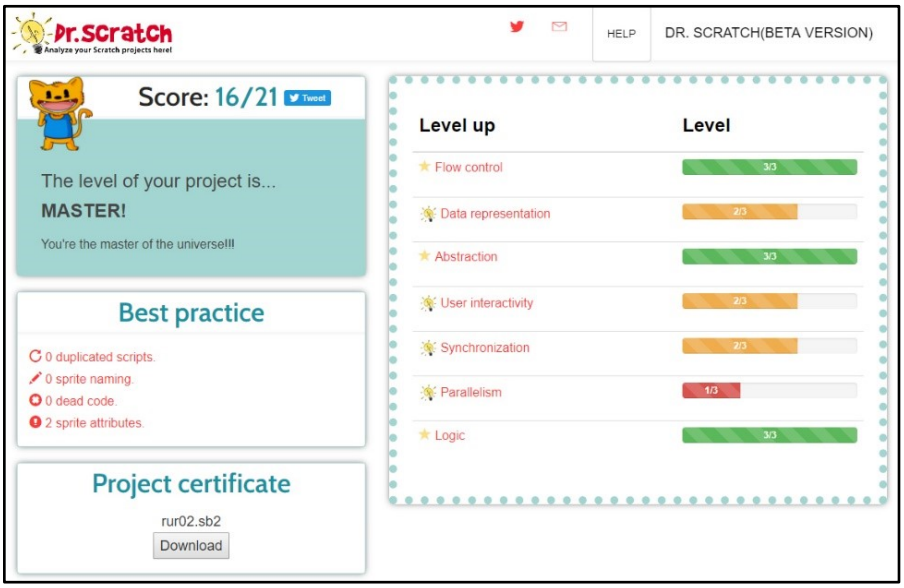
Graf 24 – Míra porozumění konceptům ve Scratch – TEST B1

4.4.2 Analýza závěrečných projektů

Analýza závěrečných projektů proběhla pomocí programu Dr. Scratch, nástroje, který hodnotí projekty Scratch z několika pohledů. Posuzuje projekty dle vlastní klasifikace konceptů informatického myšlení z hlediska využití algoritmizace, reprezentace dat, abstrakce, interaktivity, synchronizace, paralelismu a logických podmínek. V každé z těchto sedmi kategorií přiděluje program projektům nula až tři body. Maximum možných bodů je tedy 21. Podle počtu přidělených bodů analyzovaným projektům řadí projekty do tří kategorií – základní, pokročilý a odborný projekt.

- 1–6 bodů = basic – projekt základní úrovně
- 7–15 bodů = developing – pokročilý projekt
- 15–21 bodů = master – odborný projekt

Aplikace Dr. Scratch je on-line modul fungující v prohlížečích Chrome a Firefox, jehož použití je velmi snadné. Poskytuje zpětnou vazbu prostřednictvím grafů a statistik. Soubor ve formátu sb2 lze do aplikace nahrát z počítače, nebo zadat URL adresu projektu.



Obrázek 35 – Dr. Scratch

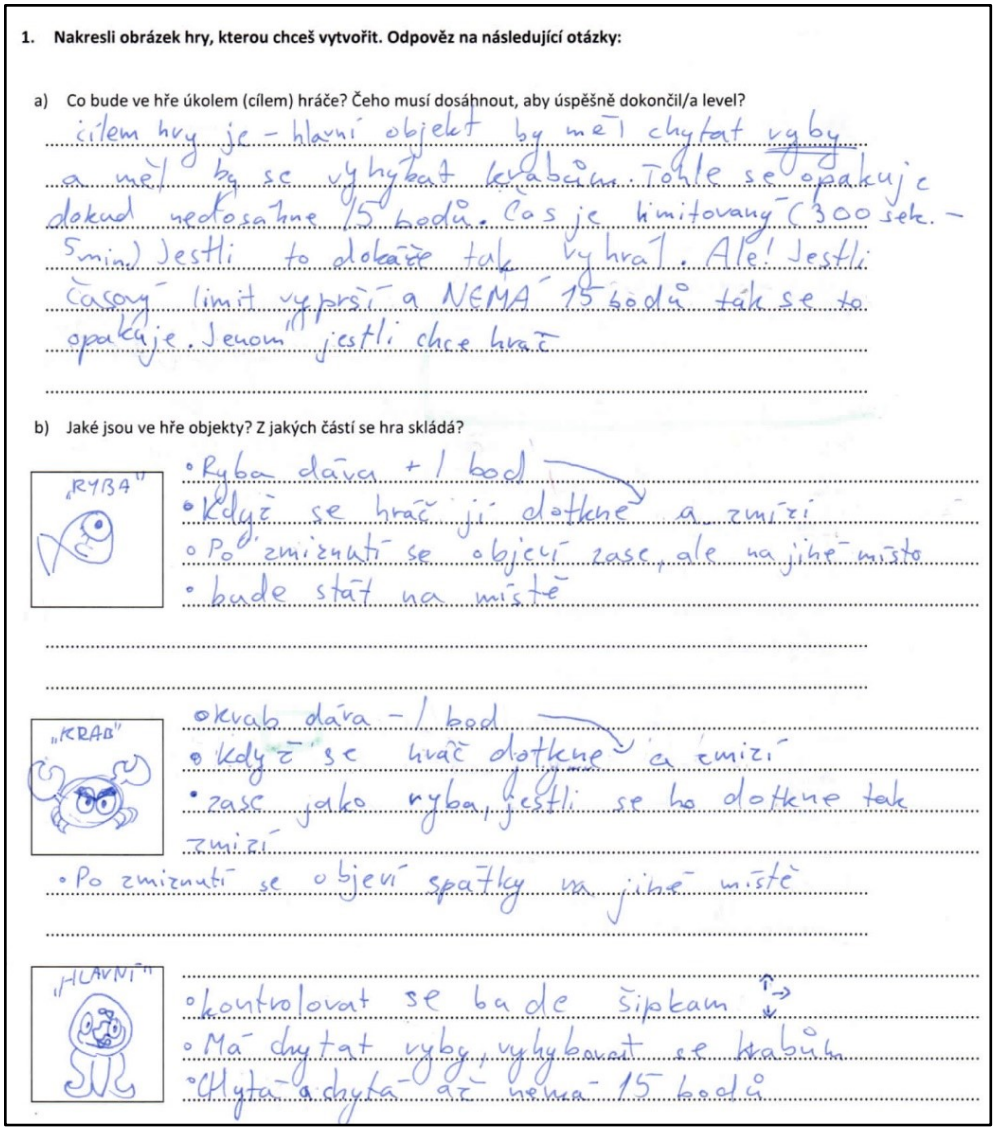
Nástroj Dr. Scratch dále kontroluje, zda v kódu nedochází k opakování scriptů, zda neobsahuje nefunkční vlákna, nepojmenované postavy, či postavy bez definovaných vlastností.

Tabulka 20 – Přiřazení bodů programem Dr. Scratch

Použité koncepty	Použité koncepty pro úroveň		
	Základní projekt	Pokročilý projekt	Odborný projekt
Algoritmizace	Sekvence	Cyklus	Cyklus s podmínkou
Reprezentace dat	Úprava vlastností objektů	Proměnná	Seznam
Abstrakce	Rozdělení skriptu na více částí	Vytvoření opakujícího se bloku příkazů	Klonování
Interaktivita	Po spuštění programu	„Po kliknutí na mne“ „Je-li stisknuta klávesa“ „Myš stisknuta“	Použití webové kamery nebo mikrofonu
Synchronizace	Synchronizace pomocí příkazu „Čekej X sek.“	Synchronizace pomocí rozesílání zpráv	Synchronizace pomocí podmínek „Po změně pozadí“ nebo „Čekej dokud“
Paralelizace	Více vláken s událostí „Po spuštění programu“	Více vláken s událostí „Po stisknutí klávesy“ Více vláken s událostí „Po kliknutí na mne“	Více vláken s událostí „Po obdržení zprávy“ Více vláken s událostí „Když startuje můj klon“ Více vláken s událostí s podmínkou „KDYŽ“
Logické podmínky	Podmínka „KDYŽ“	Podmínka „KDYŽ-TAK“	Logické operace „ANO, NE, NEBO“

Před započetím programování vlastní hry žáci podnikli řadu příprav. Jednalo se především o vlastní obrázky, které kreslili ručně v hodinách výtvarné výchovy a následně naskenovali do počítače. Následně byly vyučujícím upraveny do formátu PNG, aby bylo možné nastavit průhledné pozadí. Velmi důležitý byl i popis hry obsahující základní interakce jednotlivých objektů. Tyto přípravy se ukázaly být pro další práci žáků zásadní.

Někteří žáci, kteří své přípravy podcenili, mohli používat pouze obrázky z integrované knihovny Scratch. Také používali pouze nejjednodušší varianty programovacích konceptů a snažili se z ukázek kódů probíraných při výuce sestavit hru vlastní. Často vyžadovali pomoc a jejich hry většinou dosáhli základní úrovně. Naopak žákům, kteří své přípravy zpracovali pečlivě, byly návrhy při programování her významnou pomůckou. Tito žáci ve výrazně nižší míře požadovali při kódování rady a pomoc učitele či spolužáků a také dosáhli podstatně propracovanějších programů v kratším časovém úseku. To jim umožnilo ve zbývajícím čase hry ladit, přidávat další levely a v neposlední řadě implementovat vlastní grafiku, která jejich hry posunula na výrazně vyšší úroveň. Řada projektů tak dosáhla po vizuální i programové stránce vysoké kvality.


1. **Nakresli obrázek hry, kterou chceš vytvořit. Odpověz na následující otázky:**

a) Co bude ve hře úkolem (cílem) hráče? Čeho musí dosáhnout, aby úspěšně dokončil/a level?

cílem hry je - hlavní objekt by měl chytat ryby a měl by se vyhýbat krabům. Tohle se opakuje dokud nedosáhne 15 bodů. Čas je limitovaný (300 sek. - 5min) Jestli to dokáže tak vyhrál. Ale! Jestli časový limit vyprší a NEMÁ 15 bodů tak se to opakuje. Jenom jestli chce hrát

b) Jaké jsou ve hře objekty? Z jakých částí se hra skládá?

„RYBA"

- Ryba dává + 1 bod
- Když se hráč jí dotkne a zmizí
- Po zmizení se objeví zase, ale na jiné místo
- bude stát na místě

„KRAB"

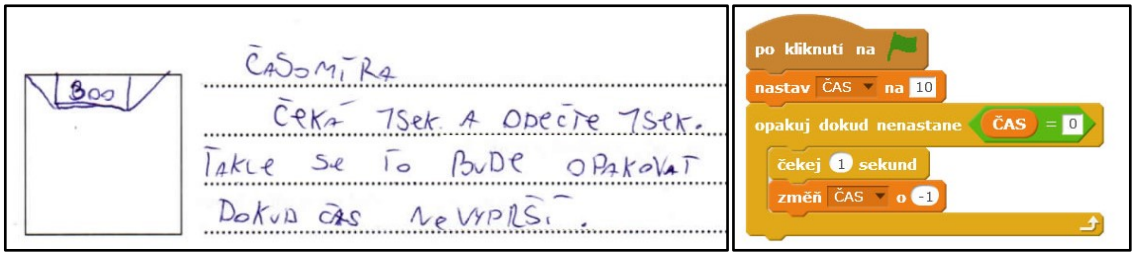
- krab dává - 1 bod
- Když se hráč dotkne a zmizí
- zase jako ryba, jestli se ho dotkne tak zmizí
- Po zmizení se objeví špatky na jiné místě

„HLAVNÍ"

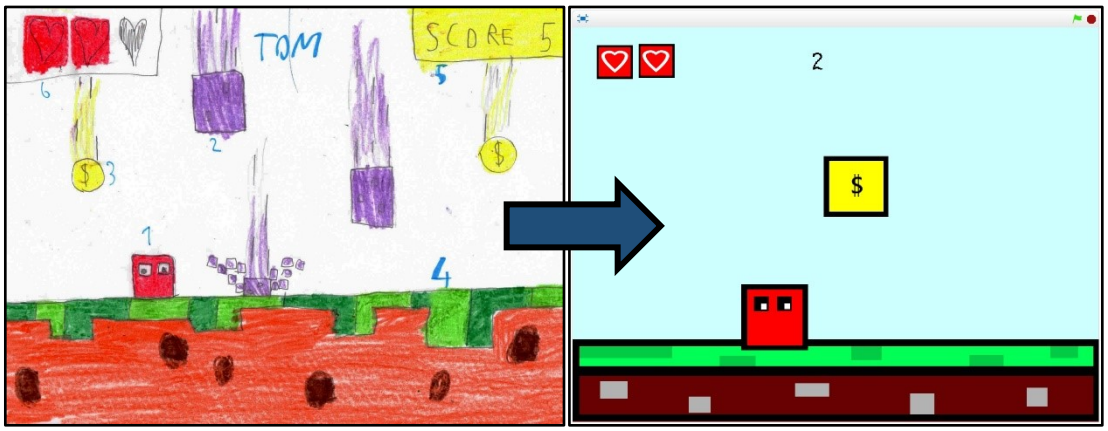
- kontrolovat se bude šipkami
- Má chytat ryby, vyhýbat se krabům
- Chyta-á chyta-á až nesmí - 15 bodů

Obrázek 36 – První část návrhu – popis objektů a jejich vlastností

Řada žáků zařadila do svých projektů i koncept proměnné. Žák č. 29 ve svém projektu velmi dobře rozepsal princip programování odpočtu časomíry ve Scratch.



Obrázek 37 – Návrh použití proměnné a výsledný kód

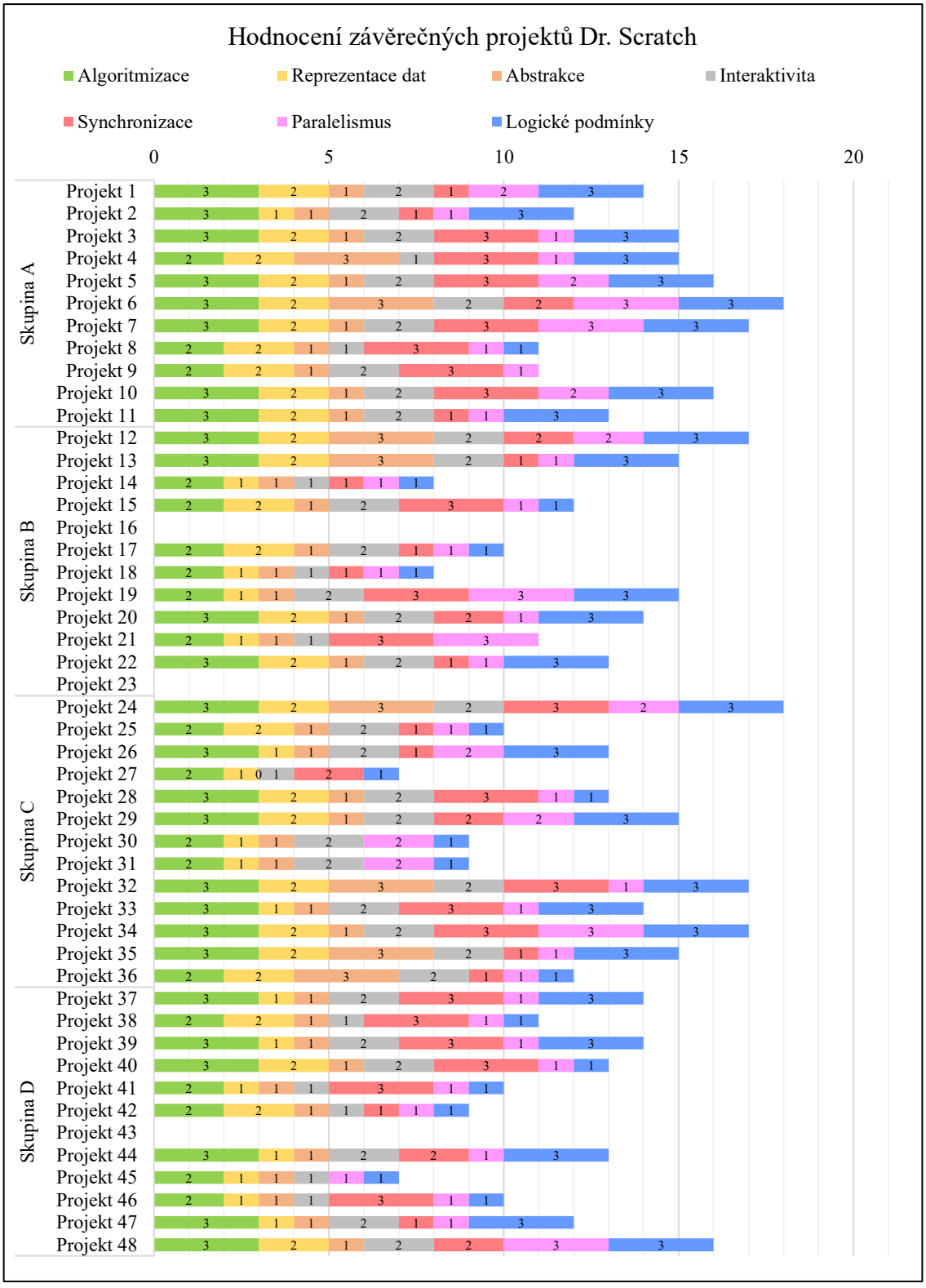


Obrázek 38 – Porovnání grafického návrhu a konečné podoby hry

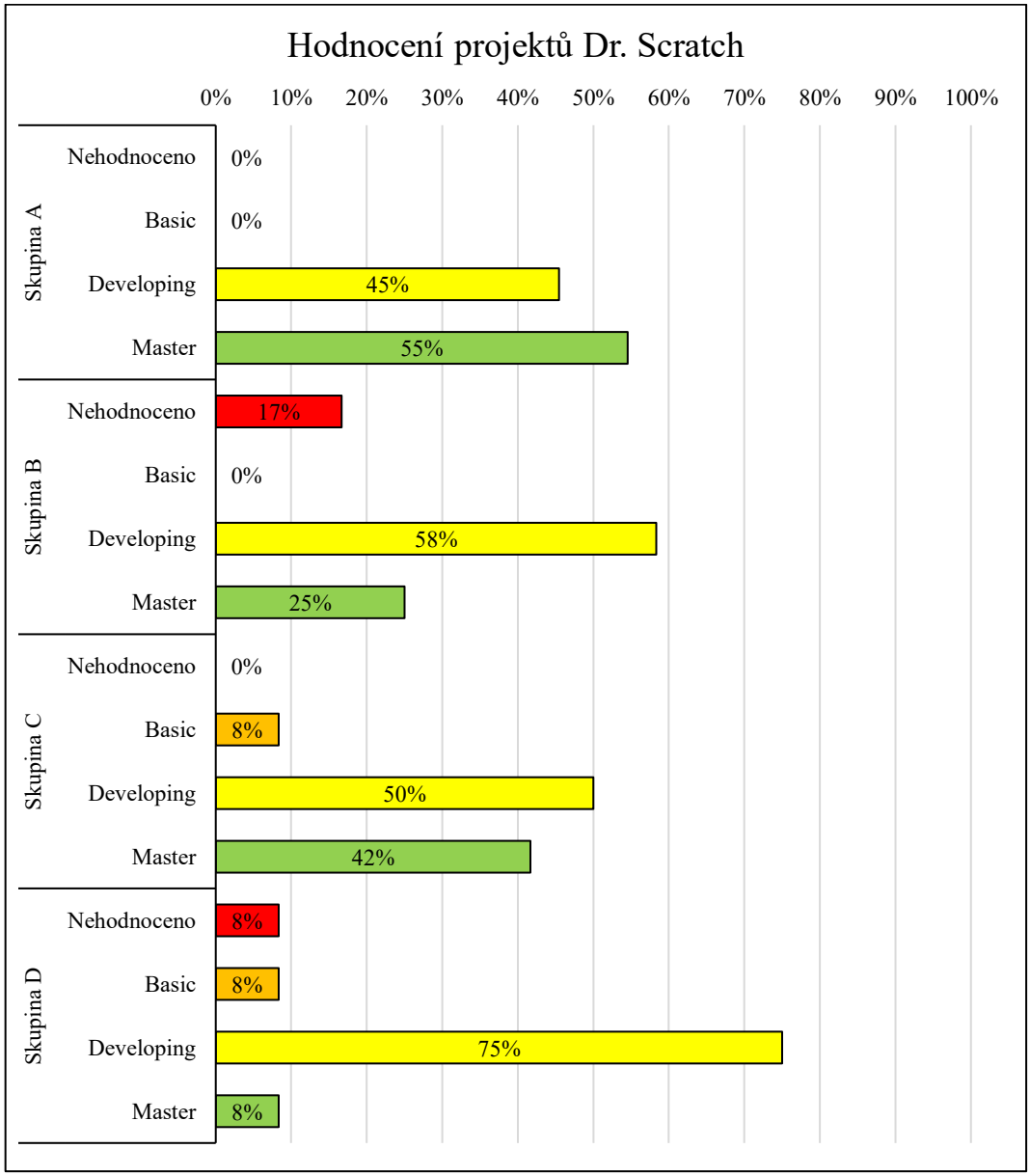
Úroveň jednotlivých projektů informatického myšlení dosáhla ve skupinách rozdílných výsledků. Z celkového počtu 48 projektů jich bylo 45 možno v programu Dr. Scratch vyhodnotit. Tři žáci nedokázali ani přes značnou dopomoc projekt dokončit do fáze, která by umožnila jejich posouzení programem Dr. Scratch.

Nejvyšší úrovně projektů dosáhli žáci skupiny A, jejichž práce byly hodnoceny v pěti případech jako pokročilé a v šesti případech jako odborně zpracované. Ve skupině B dva žáci projekt nedokončili, sedm prací dosáhlo pokročilé úrovně a tři odborné úrovně. Mezi pracemi žáků skupiny C byl pouze jeden projekt základní úrovně, šest projektů dosáhlo pokročilé úrovně a pět projektů úrovně nejvyšší, tedy odborné. Jeden žák skupiny D projekt nedokončil, jeden dosáhl úrovně základní, devět projektů bylo hodnoceno jako pokročilé a jeden jako odborný (viz Graf 26).

Závěrečné práce všech žáků skupin A, B, C i D byly analyzovány a výsledky jsou po skupinách zobrazeny v grafech č. 25 a 26.

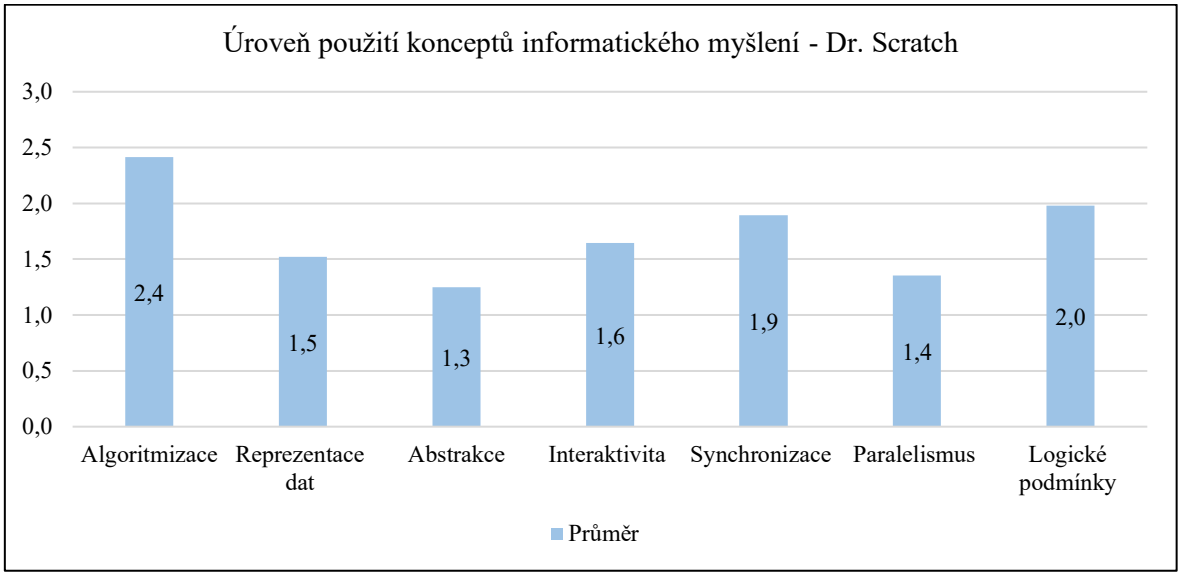


Graf 25 – Hodnocení jednotlivých projektů Dr. Scratch



Graf 26 – Hodnocení projektů Dr. Scratch

V celkovém hodnocení jednotlivých konceptů informatického myšlení dosáhli žáci nejlepších výsledků v kategorii algoritmizace a použití logických podmínek. Průměrných výsledků žáci dosáhli v případě reprezentace dat, interaktivity a synchronizace. Všeobecně nejslabších výsledků bylo dosaženo v kategoriích abstraktního myšlení a paralelizace.



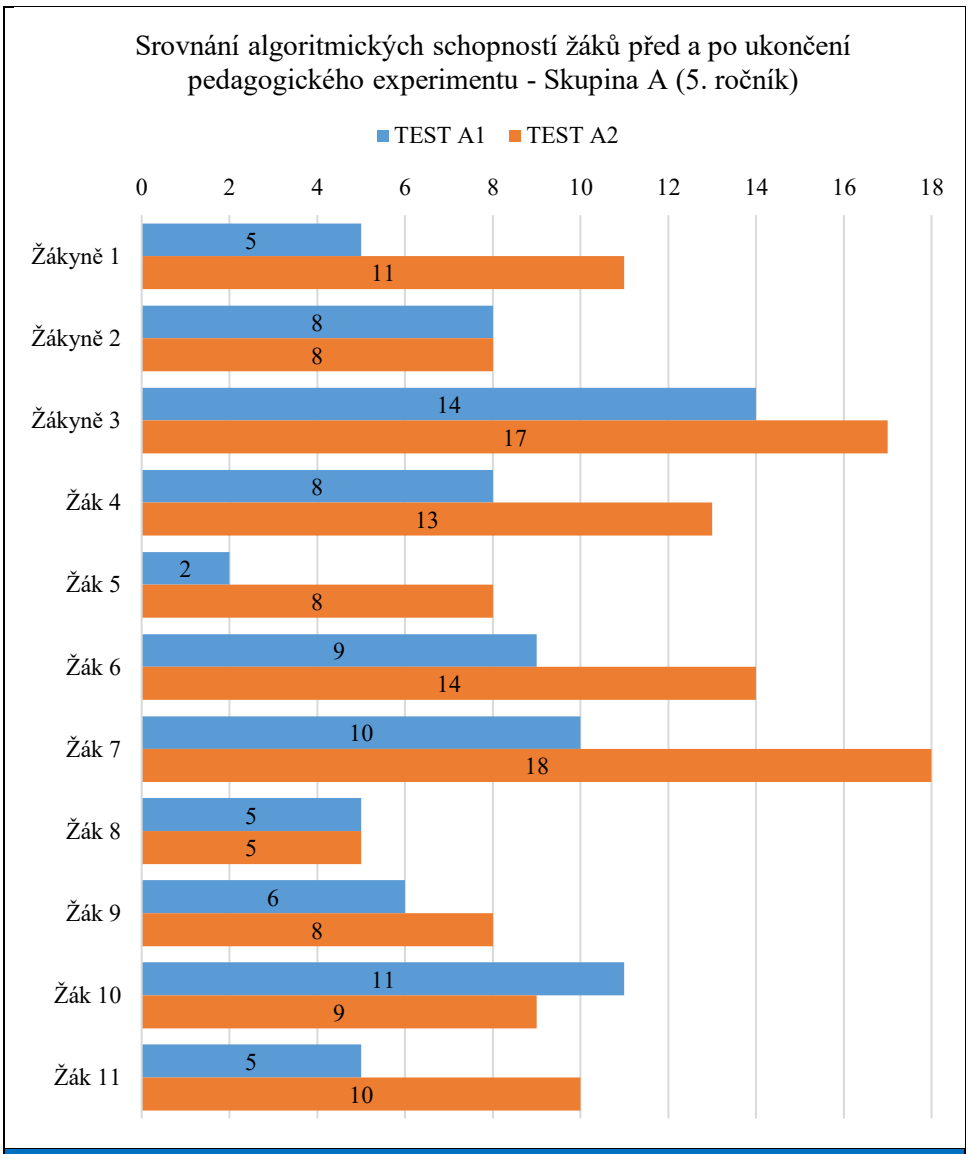
Graf 27 – Úroveň použití konceptů informatického myšlení

Velmi pozitivním zjištěním bylo dokončení 45 prací z celkového počtu 48 projektů a fakt, že žáci zvládli dokončit své hry v rámci výuky.

Pokud se žáci programování ve Scratch věnovali rovněž mimo výuku, vytvářeli své vlastní projekty, jejichž úroveň byla přiměřená pracím školním. Dva žáci, přestože před započetím experimentu neměli se Scratch žádné zkušenosti, vytvořili programy úrovně vysoce převyšující práce všech ostatních žáků.

4.4.3 Analýza rozvoje algoritmického myšlení žáků

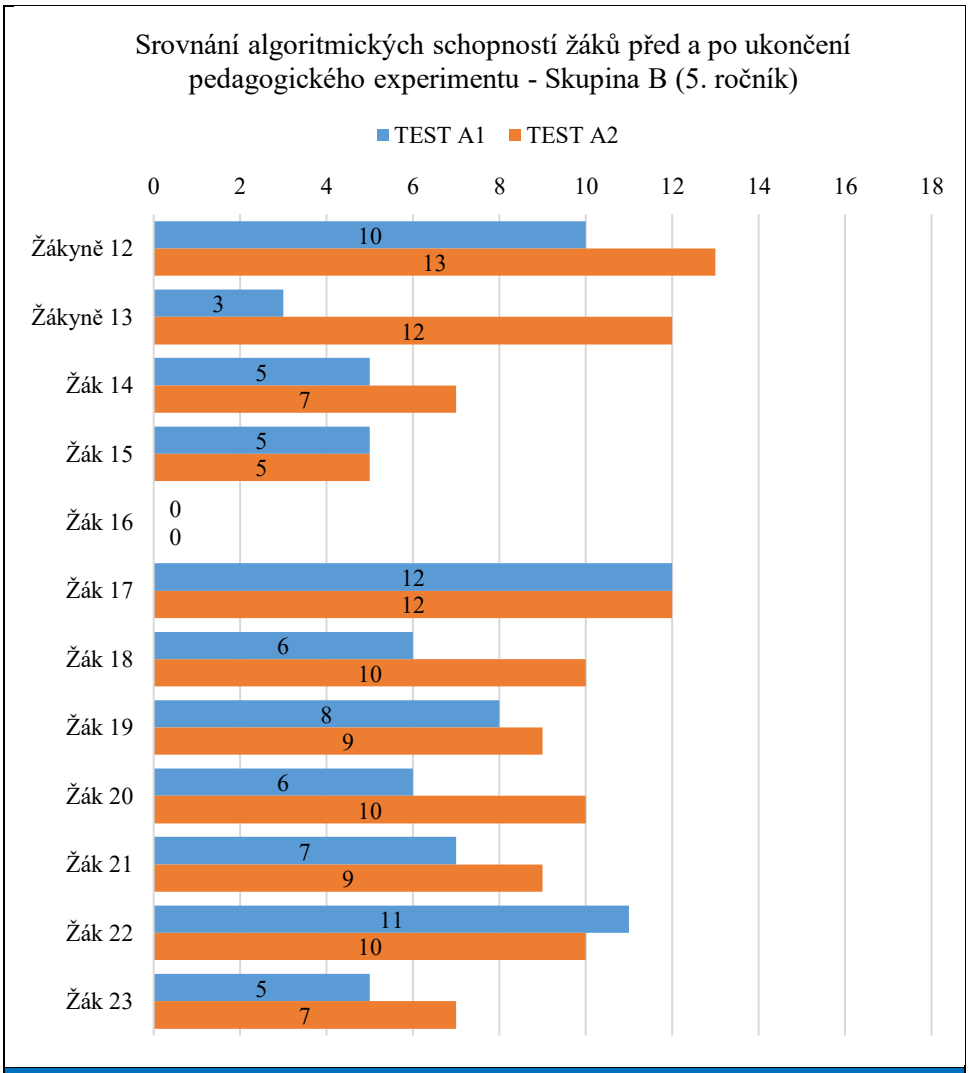
Míra rozvoje algoritmického myšlení žáků 5. a 6. třídy byla posuzována na základě výsledků testů (TEST A1, viz Příloha C a TEST A2, viz Příloha D). Časový rozdíl mezi vstupním a výstupním testováním činil sedm měsíců a porovnání výsledků obou testů dokládá značný posun v schopnostech většiny žáků řešit úlohy z oblasti algoritmizace.



Skupina A – N=11					
Test A1	Test A2	Rozdíl	Zlepšení	Stejně	Zhoršení
∅ $_1$ = 7,5 b	∅ $_2$ = 11 b	+3,5 b	8	2	1

Graf 28 – Srovnání Testu A1 a Testu A2 – Skupina A

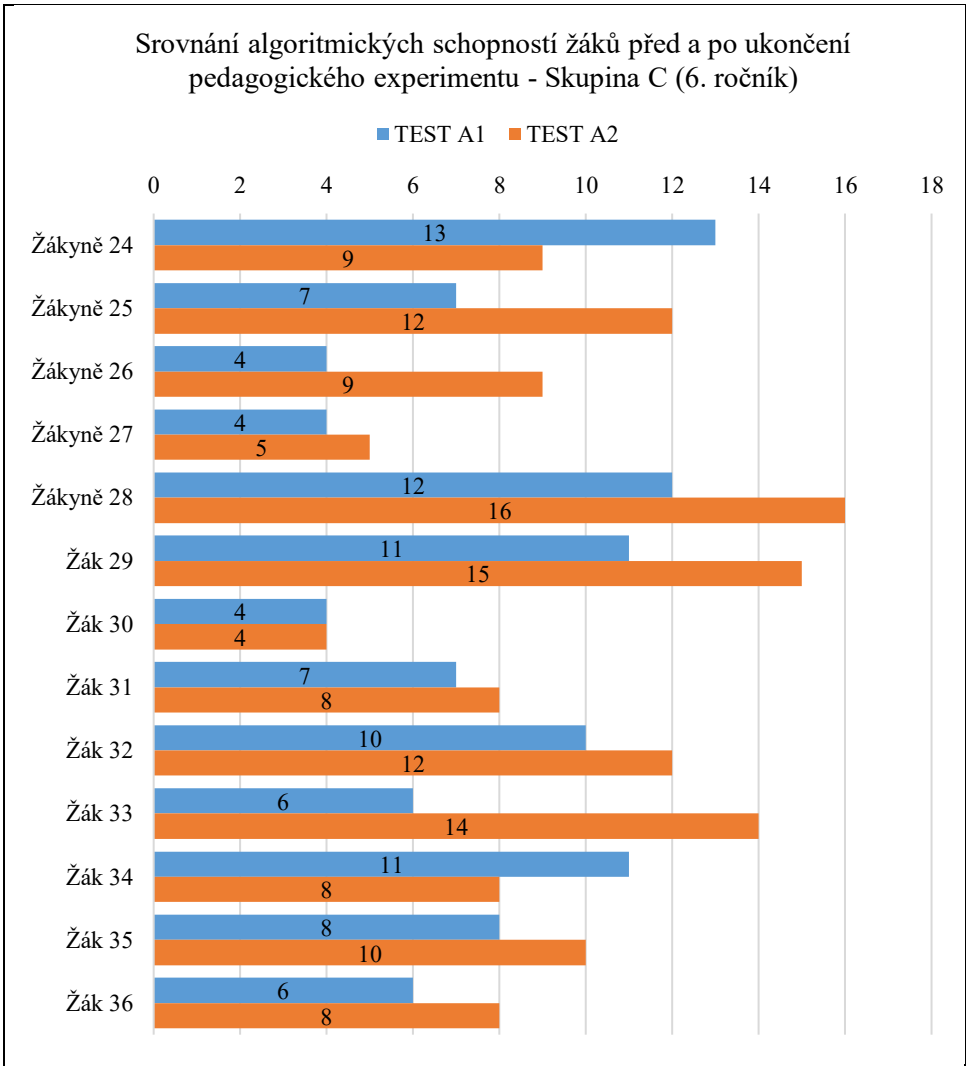
Žáci skupiny A získali ve vstupním testu průměrně 7,5 bodu. Ve výstupním testu dosáhli nejlepšího výsledku ze všech skupin, když získali průměrně 11 bodu. Z celkového počtu 11 žáků jich 8 ve výstupních testech zaznamenalo zlepšení. Dva žáci dosáhli v obou testech stejného výsledku a jeden z žáků si po ukončení experimentu pohoršil o dva body. Největší posun zaznamenal žák č. 7, který ve vstupním testu získal 10 bodů a ve výstupním zodpověděl správně všechny otázky, a dosáhl tak plného počtu 18 bodů. Průměrně si žáci skupiny A polepšili o celých 3,5 bodu (viz Graf 28).



Skupina B – N=12					
Test A1	Test A2	Rozdíl	Zlepšení	Stejně	Zhoršení
∅ $_1$ = 6,5 b	∅ $_2$ = 8,7 b	+2,2 b	8	3	1

Graf 29 – Srovnání Testu A1 a Testu A2 – Skupina B

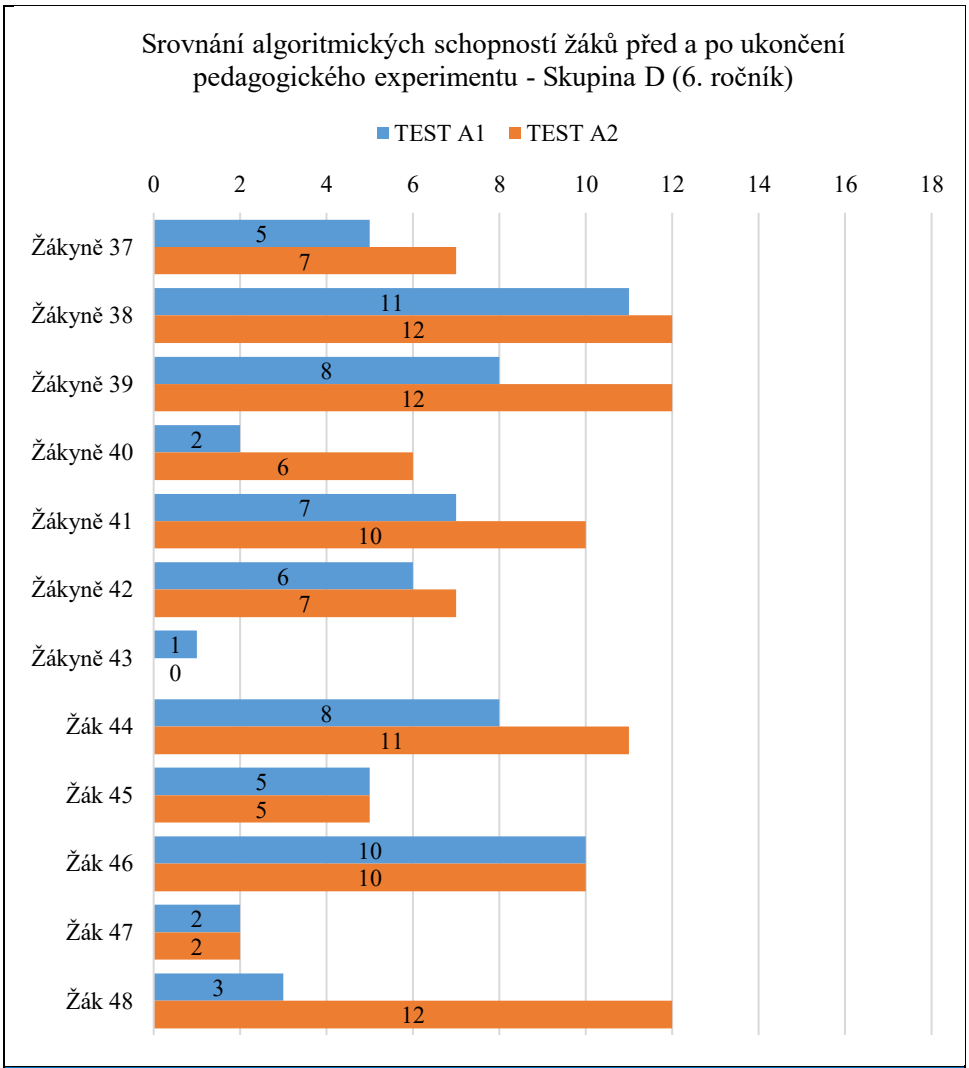
Žáci skupiny B získali ve vstupním testu průměrně 6,5 bodu a ve výstupním 8,7 bodu. Z celkového počtu 12 žáků, jich 8 po ukončení experimentu, zaznamenalo zlepšení. Tři žáci dosáhli v obou testech stejného výsledku a jeden žák si po ukončení experimentu pohoršil o jeden bod. Ve skupině se vyskytl žák, který přes fakt, že 12 z 18 otázek bylo uzavřených, nezískal v ani jednom z testů žádný bod. Největší posun zaznamenala žákyně č. 13, která ve vstupním testu získala pouze 3 body a ve výstupním odpověděla správně na dvě třetiny otázek a získala tak 12 bodů. Průměrně si žáci skupiny B polepšili o celých 2,2 bodu (viz Graf 29).



Skupina C – N=13					
Test A1	Test A2	Rozdíl	Zlepšení	Stejně	Zhoršení
∅ $_1$ = 7,9 b	∅ $_2$ = 10 b	+2,1 b	10	1	2

Graf 30 – Srovnání Testu A1 a Testu A2 – Skupina C

Žáci skupiny C získali ve vstupním testu průměrně 7,9 bodu a ve výstupním 10 bodů. Z celkového počtu 13 žáků, jich 10 ve výstupních testech zaznamenalo zlepšení, jeden žák dosáhl v obou testech stejného výsledku a dva žáci si po ukončení experimentu pohoršili. Největší posun zaznamenal žák č. 33, který ve vstupním testu dosáhl pouze 6 bodů a ve výstupním 14 bodů. Průměrně si žáci skupiny C polepšili o 2,1 bodu (viz Graf 30).



Skupina D – N=12					
Test A1	Test A2	Rozdíl	Zlepšení	Stejně	Zhoršení
∅ $_1$ = 5,7 b	∅ $_2$ = 7,8 b	+2,1 b	8	3	1

Graf 31 – Srovnání Testu A1 a Testu A2 – Skupina D

Nejslabších výsledků dosáhli žáci skupiny D, kteří získali ve vstupním testu průměrně 5,7 bodu a ve výstupním 7,8 bodu. Z celkového počtu 12 žáků jich 8 po ukončení

experimentu zaznamenalo zlepšení, tři žáci dosáhli v obou testech stejného výsledku a jedna žákyně si po ukončení experimentu pohoršila o jeden bod. Jednalo se však o případ, kdy v prvním testu získala pouhý jeden a ve druhém testu žádný bod. Největší posun zaznamenal žák č. 48, který ve vstupním testu získal pouze 3 body a ve výstupním odpověděl správně na dvě třetiny otázek a získal tak 12 bodů. Průměrně se žáci skupiny D, stejně jako žáci skupiny C, zlepšili průměrně o 2,1 bodu (viz Graf 31).

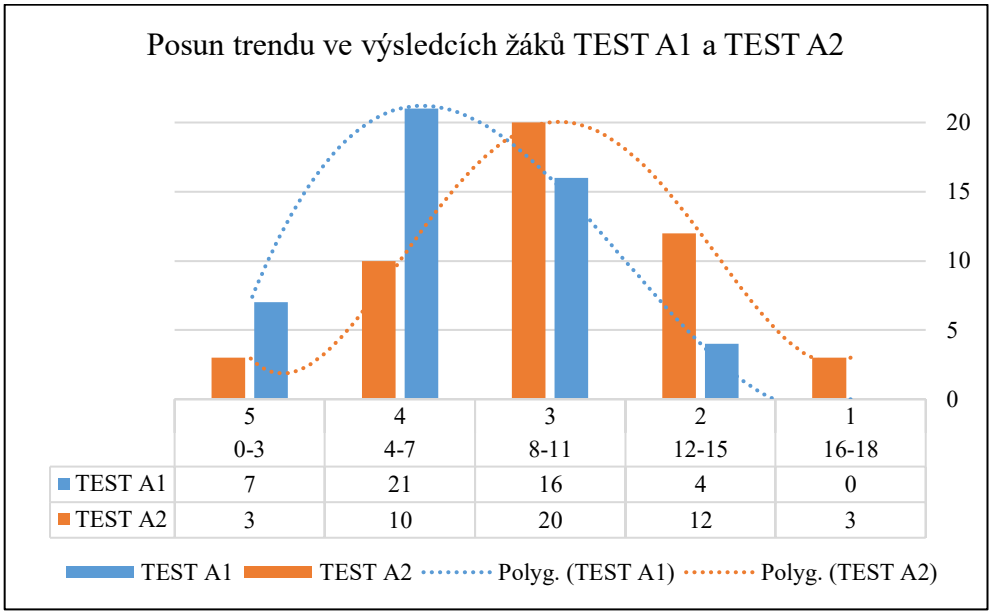
Srovnání skupin

Porovnání výsledků obou testů ukazuje, že po ukončení pedagogického experimentu si 71 % žáků v oblasti řešení úloh s algoritmickým zaměřením značně polepšilo, 19 % dosáhlo v obou testech stejného počtu bodů a 10 % žáků si ve výstupním testu pohoršilo.

Tabulka 21 – Vyhodnocení výsledků testů TEST A1 a TEST A2

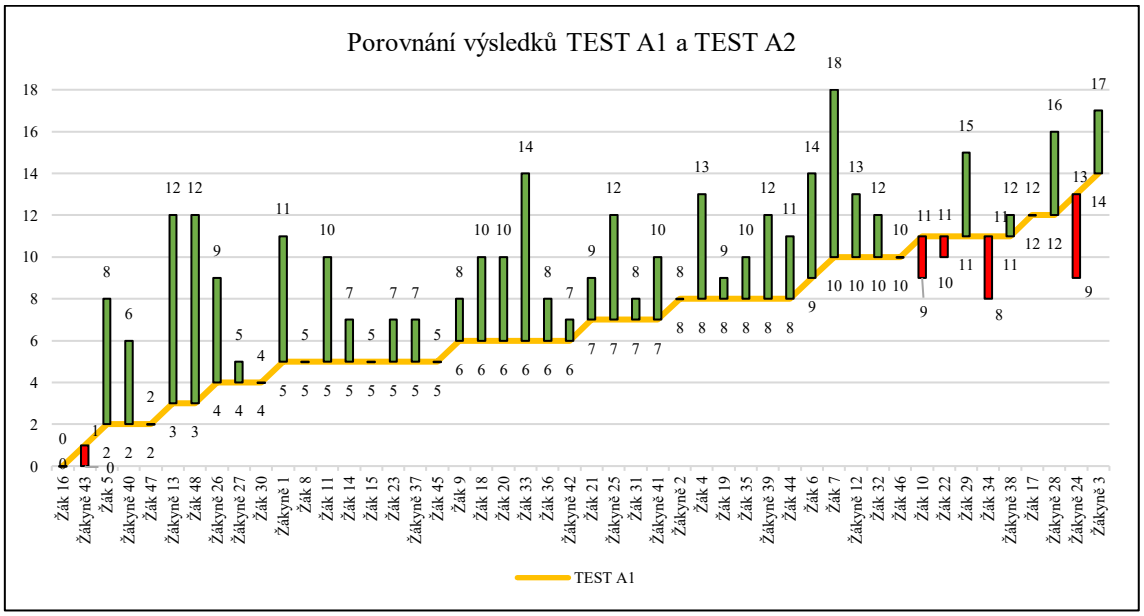
Třída	Skupina	Ø počet bodů		ø rozdíl	počet žáků, kteří dosáhli		
		TEST A1	TEST A2		zlepšení	stejného výsledku	zhoršení
5. třída	Skupina A	7,5	11	+ 3,5	8	2	1
	Skupina B	6,5	8,7	+ 2,2	8	3	1
6. třída	Skupina C	7,9	10	+ 2,1	10	1	2
	Skupina D	5,7	7,8	+ 2,1	8	3	1

Pokud bychom rozdělili výsledky žáků do pěti kategorií obdobně, jako při známkování, dostali bychom výsledky, na nichž je jednoznačně patrný posun trendu potencionálně obdržených známek (viz Graf 32).

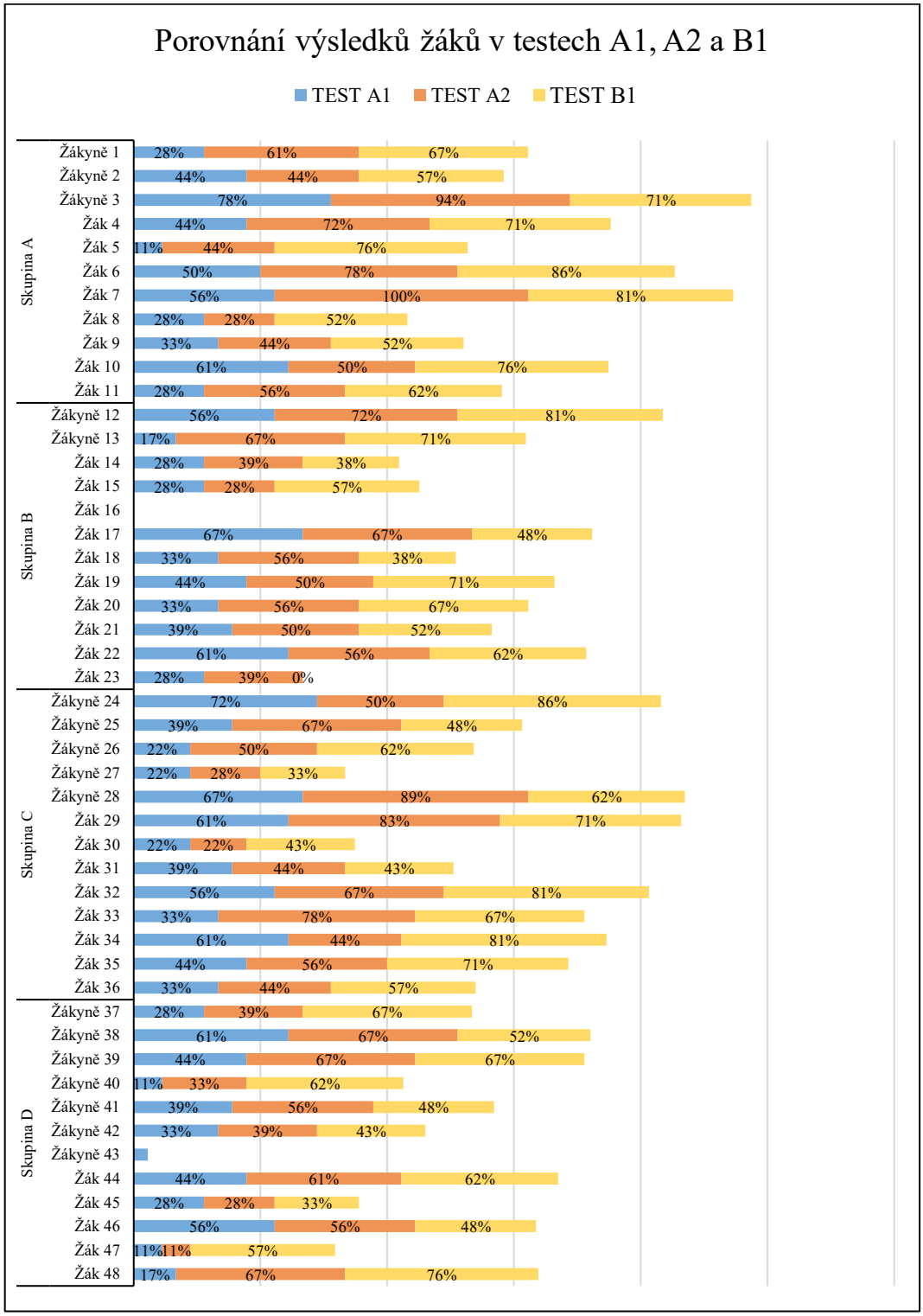


Graf 32 – Posun trendu ve výsledcích žáků TEST A1 a TEST A2

Z Grafu 33 je patrné, že lepších výsledků dosahovali spíše žáci, kteří v prvním testu (TEST A1, viz Příloha D) dosáhli maximálně deseti bodů. Mezi žáky, kteří se v prvním testu umístili na prvních devíti místech, si čtyři žáci v druhém testu pohoršili, jeden získal stejný počet bodů.



Graf 33 – Porovnání výsledků TEST A1 a TEST A2 – všichni žáci



Graf 34 – Porovnání výsledků žáků v testech – TEST A1, A2, B1

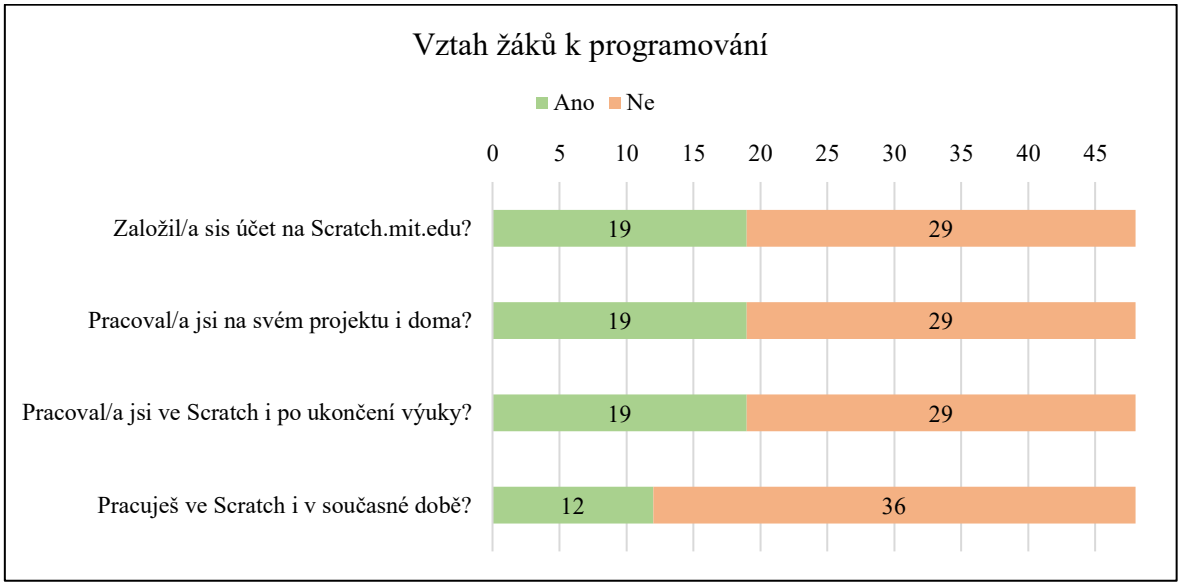
Z celkového počtu 48 žáků dosáhlo 34 žáků ve výstupním testu (TEST A2, viz Příloha E) lepších výsledků než v testu vstupním, 9 žáků získalo v obou testech stejný počet bodů a 5 žáků získalo ve druhém testu bodů méně. Celkově se tedy u 29 % žáků neprojevil v řešení úloh algoritmické povahy žádný prokazatelný progres. Vezmeme-li v potaz pouze žáky, kteří dosáhli v oblasti algoritmického myšlení zlepšení, průměrně získali na konci pedagogického experimentu o 3,75 bodu více než na jeho počátku.

Největšího pokroku dosáhli žáci 5. ročníku skupiny A, kteří získali ve druhém testu (TEST A2) ověřujícím algoritmické dovednosti v průměru o 3,5 bodu více než v testu vstupním (TEST A1). Druhého nejvyššího rozdílu dosáhli členové skupiny B, rovněž žáci 5. třídy. Jejich zlepšení činilo v průměru 2,2 bodu na jednoho žáka. Žáci skupin C a D si shodně v průměru polepšili o 2,1 bodu na jednoho žáka. Vzhledem k faktu, že výuka probíhala ve všech skupinách za obdobných podmínek, lze z výsledků testů vyvodit, že věkový rozdíl mezi žáky 5. a 6. třídy základní školy, kde pedagogický experiment probíhal, neměl na výsledky experimentu vliv. Z toho lze usoudit, že výuka základů programování v prostředí Scratch je pro žáky ve věku 10 let vhodná.

4.4.4 Závěrečný dotazník

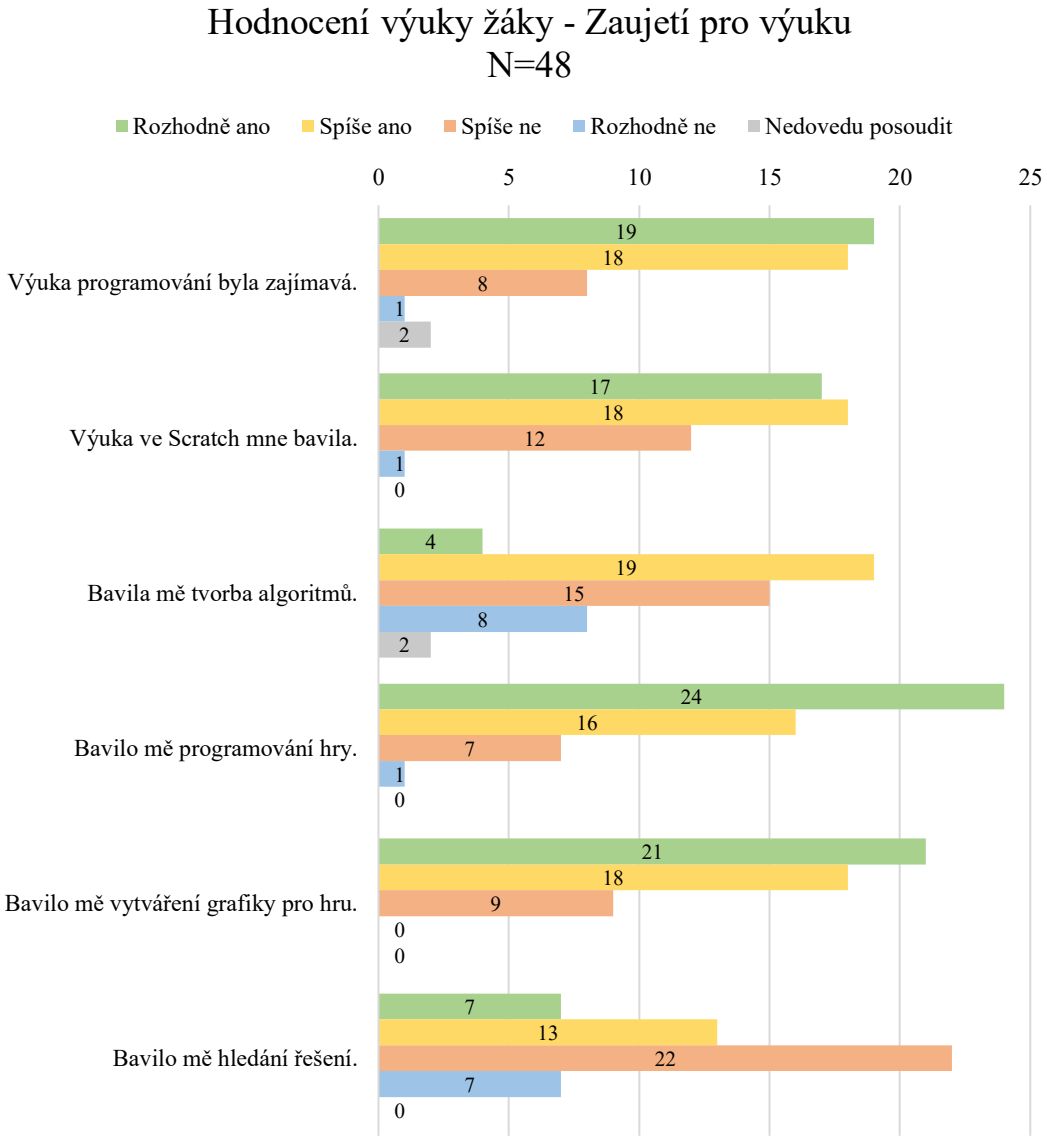
Šest týdnů po ukončení pedagogického experimentu žáci vyplnili dotazník (DOT 02), ve kterém vyjádřili svůj názor na uskutečněnou výuku a svůj vztah k programování.

Z celkového počtu 48 žáků jich 19 uvedlo, že si dobrovolně založili vlastní účet na portálu Scratch.org a pracovalo na vlastních projektech i mimo vyučování. Rovněž 19 žáků uvedlo, že výuka základů programování je oslovila natolik, že pokračovali v programování v Scratch i po ukončení výuky. Z tohoto počtu 12 žáků uvedlo, že pokračují v aktivitách se Scratch i nadále. Na otázku zda budou v práci se Scratch pokračovat i nadále uvedlo 7 žáků odpověď „Rozhodně ano“ a dalších 13 žáků „Spíše ano“.



Graf 35 – Vztah žáků k programování

Z odpovědí na otázky týkajících se zajímavosti výuky vyplývá, že většinu žáků výuka zaujala. Za nejzábavnější označili žáci práci na závěrečném projektu. Na prvním místě činnost samotného programování vlastní hry a na místě druhém tvorbu grafiky pro hru. Tvorbu algoritmů žáci hodnotili neutrálně a za nejméně zábavné považují žáci odhalování problémů a hledání řešení. V slovním hodnocení se žáci často vyjadřovali také k testování her, kdy jako velmi oblíbenou činnost uváděli testování vlastních her a her spolužáků a jejich vzájemné porovnávání.

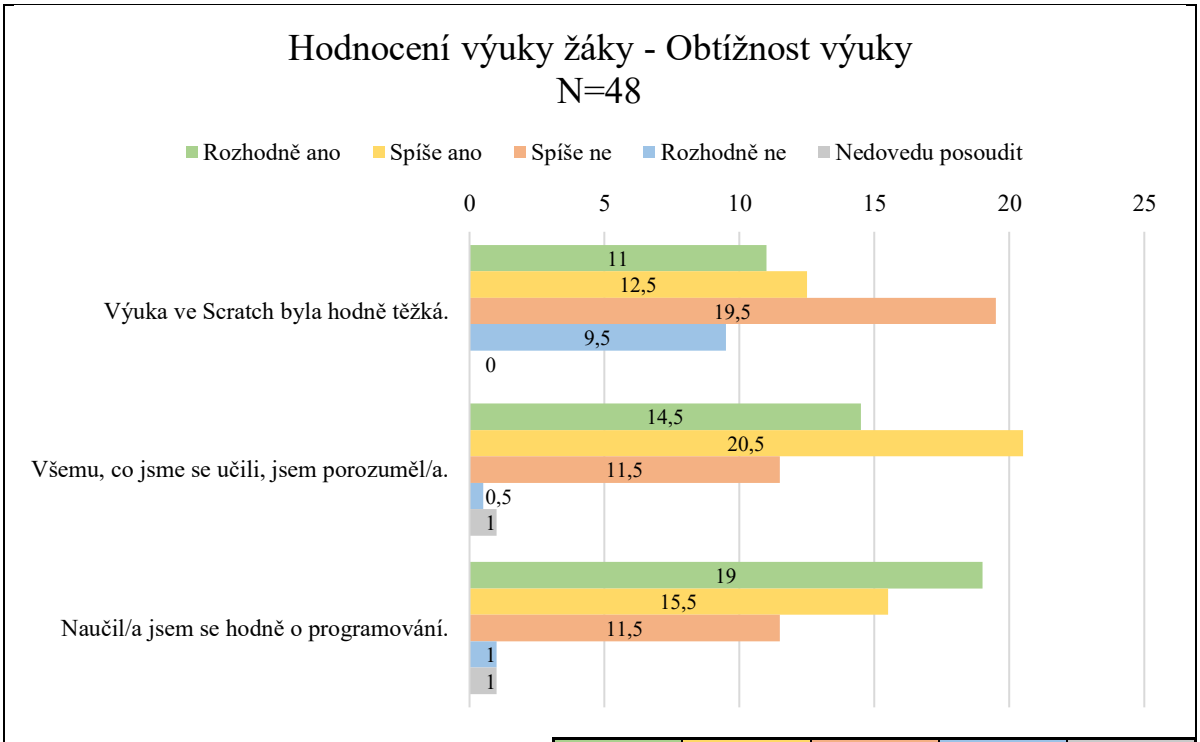


	Rozhodně ano	Spíše ano	Spíše ne	Rozhodně ne	Nedovedu posoudit
Výuka programování byla zajímavá.	19	18	8	1	2
Výuka ve Scratch mne bavila.	17	18	12	1	0
Bavila mě tvorba algoritmů.	4	19	15	8	2
Bavilo mě programování hry.	24	16	7	1	0
Bavilo mě vytváření grafiky pro hru.	21	18	9	0	0
Bavilo mě hledání řešení.	7	13	22	7	0

Graf 36 – Hodnocení výuky žáky – Zaujetí pro výuku

Výuku základů programování žáci nejčastěji vnímali jako spíše snadnou. Celkem 36 žáků se domnívá, že probíranému učivu porozuměli, z toho 13 žáků na tvrzení „Všemu, co jsme

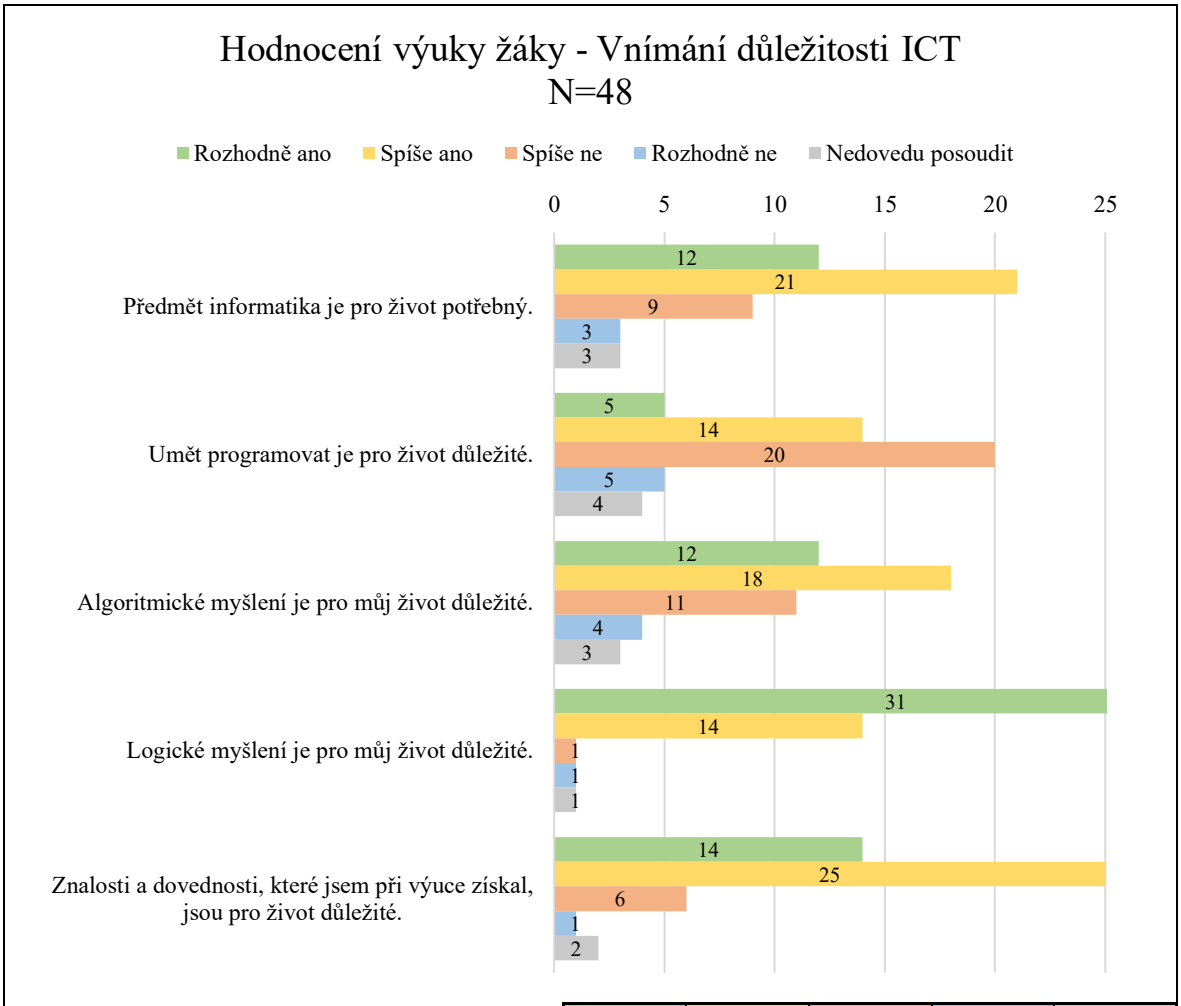
se učili, jsem porozuměl/a“ odpovědělo „Rozhodně ano“ a 23 žáků „Spíše ano“. Současně se žáci přiklání k faktu, že se o programování hodně naučili.



	Rozhodně ano	Spíše ano	Spíše ne	Rozhodně ne	Nedovedu posoudit
Výuka ve Scratch byla hodně těžká.	11	12,5	19,5	9,5	0
Všemu, co jsme se učili, jsem porozuměl/a.	14,5	20,5	11,5	0,5	1
Naučil/a jsem se hodně o programování.	19	15,5	11,5	1	1

Graf 37 – Hodnocení výuky žáky – Obtížnost výuky

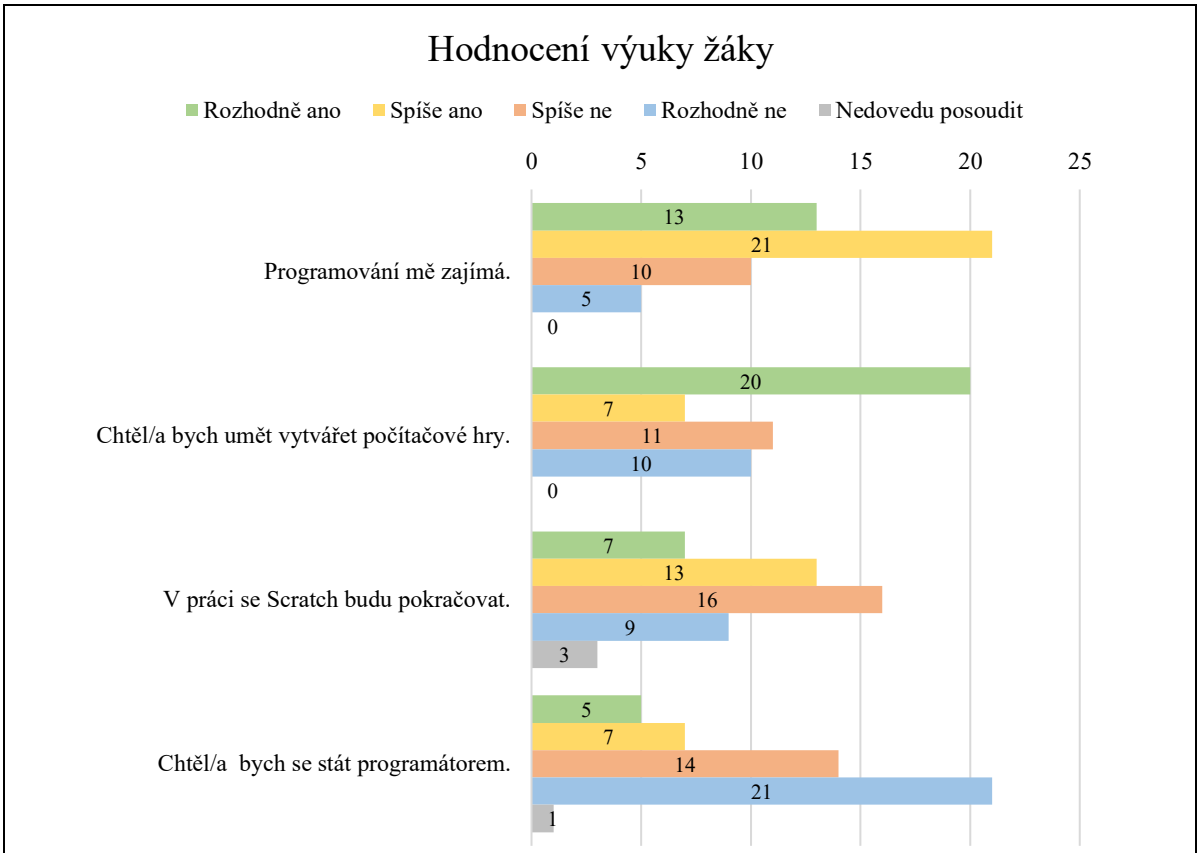
Žáci vnímají výuku předmětu Informatika jako důležitou. Celkem 12 žáků zvolilo u tvrzení, že „Předmět Informatika je pro život potřebný“ možnost „Rozhodně ano“ a 21 žáků „Spíše ano“. Jako nejvíce potřebné pro život žáci nejčastěji označili logické myšlení, které celých 31 žáků označilo jako „rozhodně důležité“ a 14 žáků jako „spíše důležité“. Schopnost algoritmického myšlení však jako důležitou pro život vidí již jen 30 žáků. Umět programovat většina žáků považuje za nedůležité. Přesto se 39 žáků domnívá, že znalosti a dovednosti, které během výuky základů programování získali, jsou pro život důležité.



	Rozhodně ano	Spíše ano	Spíše ne	Rozhodně ne	Nedovedu posoudit
Předmět informatika je pro život potřebný.	12	21	9	3	3
Umět programovat je pro život důležité.	5	14	20	5	4
Algoritmické myšlení je pro můj život důležité.	12	18	11	4	3
Logické myšlení je pro můj budoucí život důležité.	31	14	1	1	1
Znalosti a dovednosti, které jsem při výuce získal, jsou pro život důležité.	14	25	6	1	2

Graf 38 – Hodnocení výuky žáky – Vnímání důležitosti ICT

Na počátku pedagogického experimentu neměl nikdo z žáků zkušenost s prostředím Scratch a pouze 5 žáků uvedlo, že již v minulosti programovali. V závěrečném dotazníku, na otázku zda budou žáci s programováním ve Scratch pokračovat, 7 žáků uvedlo, že „Rozhodně ano“ a 13 žáků, že „Spíše ano“. Posun ve vztahu žáků k programování dokládá i fakt, že po dokončení experimentu ji 34 uvedlo, že je programování zajímá. Celkem 27 žáků se vyjádřilo ve smyslu, že by rádi uměli vytvářet počítačové hry, 5 žáků by se rozhodně chtělo stát programátory a 7 žáků tuto možnost zvažuje.



	Rozhodně ano	Spíše ano	Spíše ne	Rozhodně ne	Nedovedu posoudit
Programování mě zajímá.	13	21	10	5	0
Chtěl/a bych umět vytvářet počítačové hry.	20	7	11	10	0
V práci se Scratch budu pokračovat.	7	13	16	9	3
Chtěl/a bych se stát programátorem.	5	7	14	21	1

Graf 39 – Hodnocení výuky žáky – Vztah k programování

V druhé části dotazníku měli žáci nejprve zodpovědět na otázku, kde získali nejvíc svých počítačových dovedností. Školu nebo učitele na prvním místě uvedlo 22 žáků a dalších 21 žáků na místě druhém. Pouze dva žáci neuvedli školu či učitele vůbec. Jako další zdroje svých počítačových dovedností uváděli žáci nejčastěji rodinné příslušníky, v několika málo případech označili sami sebe a ve dvou případech internet.

Ve druhé otázce žáci měli napsat, co si myslí o výuce programování ve Scratch: jak je bavila, co by na výuce změnili, doplnili či naopak zrušili. Nejpozitivněji žáci hodnotili práci na závěrečných projektech. Jednoznačně nejčastěji bylo zmiňováno programování vlastní hry a na druhém místě tvorba grafiky.

„Nejvíc mě bavilo tvoření hry a vtipů. Mohli jsme i nahrávat zvuk.“

„Výuka mě docela bavila, protože jsem zjistil, že to je docela zvláštní, co počítač dokáže udělat tak jednoduše. Nedoplnil bych nic, protože toho bylo akorát.“

„Výuka mě bavila jen trochu. Dobrý bylo tvoření hry a ta grafika. Radši bych ale dělala prezentace nebo rozvrhy ve Wordu.“

Někteří žáci 6. třídy uváděli, že je výuka nebavila, protože byla moc těžká a ne všemu porozuměli. Především tito žáci navrhovali přidat do Scratch další příkazy.

„Moc mě Scratch nebavil. Podle mě je Scratch až moc jednoduchý pro děti v 6. třídě. Je tam málo možností. Přidal bych například „Vyskoč“, „Utíkej“ atd. Chtělo by to jiný program.“

„Výuka programování mě nebavila, protože mi to naprosto nešlo. Kdo má rozumět těm příkazům. Měli by se tam dát jiný. Třeba „Jdi do pravýho horního rohu“, nebo „Skákej“. Bavila mě grafika, to byla docela zábava. A taky algoritmy mě bavily, ty byly zábava.“

Před započetím přípravy jednotlivých projektů měli žáci možnost zvolit, zda chtějí pracovat jednotlivě či ve dvojicích. Všichni žáci se rozhodli pro samostatnou práci. Zhruba polovina těch žáků, kteří se v závěrečném dotazníku k tomuto faktu vyjádřili, si samostatnou práci pochvalovali, protože měli na programování klid. Druhá polovina se zmínila, že by raději pracovali ve dvojicích. Jako důvod uváděli možnost se častěji mezi sebou radit (přestože tuto možnost žáci měli), a také by to „bylo zábavnější“.

„Programování ve Scratchi je super. Změnil bych, abychom pracovali týmově. Učím se něco nového a to mě těší.“

„Scratch mě bavil, protože jsem mohl rozvinout fantazii a mohli jsme sdílet a prohlížet si jiné projekty.“

„Bylo super, že jsem dělala sama, ale když jsem něco nerozuměla, tak jsem se mohla zeptat spolužáků, a když to nevěděli tak učitele.“

Některým žákům se výuka zdála příliš jednoduchá:

„Scratch mě velice bavil, ale pokavaď chci být programátor, musím se posunout dál. Zrovna teď bych změnil programování ve Scratchi a dal bych tam GameMaker 8.0.“

„Scratch je dobrý. Naučí vás programovat základy. Podle mě je ale Scratch pro začátečníky, aby se naučili programovat. Já radši pracuju s enginy. Například Gamemaker 8 nebo Uniti 5. Taky programuju v jazyce Javascripty, CSS a HTML.“

Velmi potěšující byla hodnocení od žáků, kteří před výukou neměli s programováním žádnou zkušenost a v závěrečném dotazníku se vyjadřovali všeobecně pozitivně.

„No já jsem na začátku informatice vůbec nerozuměl a spíš jsem se nechtěl učit o programování. Ne, že by mě informatika nezajímala. Je důležité mít všeobecný přehled, ale mě to prostě na počítači nebaví. To programování mě bavilo, když jsem pak věděl, jak to mám udělat. Podle mě není žádný problém v informatice, ale ve mně. Nebaví mě sedět u počítače (doma žádný nemám), takže jsem byl celkem smutný, že jsem se na to doma podívat nemohl. Ale vyučování dávám 1 ✶.“

Na závěr dotazníku měli žáci nakreslit jednoduchý obrázek, který by vyjadřoval jejich vztah k programování ve Scratch. Také v tomto případě většina obrázků vyjadřovala pozitivní vztah žáků k programování ve Scratch, i když se vyskytly i případy, kdy žáci vyjádřili nezájem o výuku nebo dokonce postoj zcela negativní.

Obrázek 39 – Vztah žáků k programování – Obrázky ze závěrečných dotazníků

Pozitivně výuku hodnotili častěji žáci 5. třídy, především žáci skupiny A. Ti dosáhli druhých nejlepších průměrných výsledků ve vstupních testech a nejvyššího průměrného pokroku v testu výstupním. Také jejich projekty byly co do grafické podoby i z hlediska míry použitých programovacích konceptů nejpropracovanější.

4.4.5 Vyvození závěrů

Cílem pedagogického experimentu bylo prostřednictvím výuky základů programování ve Scratch zjistit, zda lze prostřednictvím aktivit ve Scratch přispět k rozvoji algoritmického myšlení žáků ZŠ. Při analýze dopadu výuky bylo využito především metod soustavného pozorování, rozhovorů se žáky, analýzy výsledných prací žáků a testování schopnosti algoritmizace. Následně byly vyvozeny závěry, které umožnily odpovědět na výzkumné otázky.

Otázka 1: Jaký mají žáci skupin zařazených do experimentu s využitím Scratch vztah k programování?

Z celkového počtu 48 žáků nemají mimo školu možnost pracovat na počítači pouze dva. Všichni ostatní žáci, kteří se účastnili pedagogického experimentu, mají již zkušenosti s používáním počítače. Dvě třetiny rodičů žáků kontrolují a omezují čas, který děti práci na počítači věnují. Přesto množství času, který žáci tráví u počítačů, převyšuje doporučení odborníků trávit v této věkové kategorii u počítače maximálně dvě hodiny denně, neboť průměrně žáci využívají počítač šest dní v týdnu, nejčastěji 3–4 hodiny denně. Alarmující je rovněž fakt, že pouze 26 % rodičů kontroluje činnost, kterou děti na počítači vykonávají. Nejvíce času, žáci tráví hraním počítačových her. Dále pak především brouzdáním po internetu a komunikací pomocí sociálních sítí. Tyto činnosti tvoří celých 79 % času, který žáci počítačům věnují. Příprava do školy a tvořivé činnosti, které jsou pro děti tohoto věku velmi důležité, se na činnostech vykonávaných na počítači podílejí pouhými 21 % času.

Celkem 61 % dětí pochází z rodin, kde rodiče využívají počítač pro své povolání, z toho 9 % žáků z rodin, kde alespoň jeden z rodičů je zaměstnán jako programátor. Pouze 11 % žáků vlastní programovatelné hračky (nejčastěji se jedná o robotické stavebnice) a 14 % žáků má osobní zkušenost s programováním. Před započetím experimentu neměl žádný z žáků zkušenost s programováním ve Scratch.

Rozvíjí výuka ve Scratch u žáků 5. a 6. ročníku ZŠ algoritmické myšlení?

Součástí pedagogického experimentu bylo uskutečnit šetření s cílem zjistit, zda výuka zaměřená na základy programování a aktivity se Scratch přispívají nejen k osvojení nových vědomostí a dovedností v oblasti programování, ale také k rozvoji algoritmického myšlení žáků a jejich kompetencí v oblasti řešení problémů. Tato skutečnost byla zkoumána na základě porovnání rozdílů mezi výsledky vstupních a výstupních testů (TEST A1 a TEST A2). Šetření prokázalo významný rozdíl mezi výsledky žáků při řešení úloh algoritmické povahy na počátku a na konci experimentu. Z celkového počtu 48 žáků zařazených do experimentu jich 34 zaznamenalo při výstupním testování lepších výsledků než před započetím výuky. Pouze 9 žáků dosáhlo stejných výsledků v obou testech a pět žáků zaznamenalo lepší výsledky při testování vstupním.

Skutečnost, že aktivity se Scratch přispívají k rozvoji algoritmického myšlení, potvrzují také výsledky hodnocení závěrečných projektů pomocí nástroje Dr. Scratch. Ze všech sedmi kategorií konceptů informatického myšlení, které Dr. Scratch hodnotí, vyniká na prvním místě s nejvyšším počtem získaných bodů právě koncept algoritmizace.

Na základě rozdílů mezi oběma testy a rozboru závěrečných projektů lze tedy vyvodit závěr, že realizovaná výuka a aktivity se Scratch mají pozitivní vliv na algoritmické myšlení žáků a jejich schopnost řešit úlohy algoritmické povahy.

Otázka 2: Zvýší výuka základů programování ve Scratch u žáků zájem o programování?

Na počátku pedagogického experimentu pouze 5 žáků z celkového počtu 48 uvedlo, že v minulosti již programovali. Všichni v aplikaci Game Maker. S aktivitami ve Scratch neměli žáci před započetím výuky žádné zkušenosti.

Na počátku výuky bylo žákům představeno několik jednoduchých her vytvořených v prostředí Scratch. Ty byly založeny na základních programovacích konceptech a byl u nich předpoklad, že jim podobné hry budou žáci po ukončení první fáze výuky schopni samostatně vytvářet. To mělo za cíl žáky motivovat, ale současně také eliminovat případné nereálné představy o jejich výsledných projektech a zabránit tak budoucímu zklamání.

Výsledky dotazníku (DOT 02) naznačují, že právě tvorba vlastních počítačových her byla pro žáky největší motivací. Z celkového počtu 48 žáků jich 16 uvedlo, že si dobrovolně

založili vlastní účet na portálu Scratch. Protože 34 žáků uvedlo, že je programování zaujalo, 20 žáků hodlá v programování se Scratch pokračovat a 12 zvažuje možnost stát se v budoucnosti programátorem, lze dojít k závěru, že po ukončení pedagogického experimentu je zájem o programování vyšší než před jeho započetím.

Otázka 3: Bylo by vhodné zařadit výuku programování ve Scratch do obsahu kurikulárních dokumentů?

Výuka byla naplánována a realizována ve dvou fázích. V první fázi se žáci učili jednotlivým programovacím konceptům, skládali jednoduché algoritmy a řešili zadaná cvičení ve Scratch. V druhé fázi vytvářeli v prostředí Scratch vlastní počítačovou hru. V průběhu experimentu i ve výsledcích závěrečného dotazníku se ukázalo, že právě vidina naprogramování vlastní počítačové hry byla pro žáky největší motivací. Oproti tomu výuka jednotlivých programovacích konceptů se žákům jevila jako méně zajímavá, přestože se jednalo především o aktivní hledání řešení obdobných problémů, se kterými se pak žáci potýkali při tvorbě závěrečného projektu. Z tohoto důvodu by bylo vhodnější po celou dobu výuky pracovat na jednom komplexnějším projektu, rozděleném na menší, dílčí úlohy, ve kterých by se žáci průběžně učili jednotlivým programovacím konceptům. Osvojené dovednosti při práci ve Scratch by tak ihned využívali pro zakomponování do svých projektů, například počítačových her, přičemž by se snížila časová náročnost výuky.

Během pedagogického experimentu vyvstala celá řada problémů, které přímo souvisí s obsahem výuky v dalších předmětech, především v matematice. Žáci ve věku 10–12 let nemají zpravidla zkušenosti se souřadnicovým systémem, zápornými čísly, proměnnými, členěním kruhu na stupně, neznají pojem úhel. Toto učivo je na škole, kde probíhal pedagogický experiment, probírána až od šestého ročníku výše. Proto některé prvky výuky byly i pro žáky 6. ročníku novinkou a bylo tedy nutné se v rámci experimentu zaměřit i na výuku učiva z oblasti matematiky. Z tohoto důvodu je forma výuky, která byla zvolena pro pedagogický experiment, přiměřená spíše žákům druhého stupně základní školy a je vhodné ji v případě 6. ročníku zařadit až do druhého pololetí, kdy je již značná část zmíněného učiva v matematice probrána.

Pro žáky 5. ročníku jsou navíc neznámé i principy bitmapové a vektorové grafiky, které prostředí Scratch používá. Proto je třeba v případě zařazení základů programování do výuky

snížit obtížnost probíraného učiva na takovou, která odpovídá schopnostem žáků tohoto věku, aby osvojení znalostí a dovedností probíhalo přirozeným způsobem. Výsledky testů však ukazují, že i přes komplikace v podobě neznalosti některých prvků měla výuka v prostředí Scratch pro většinu žáků, včetně žáků 5. ročníku, v oblasti rozvoje algoritmických dovedností značný přínos; v případě skupiny A žáci dokonce dosáhli lepších výsledků než žáci 6. třídy.

Současný RVP ZV však problematiku programování nezmiňuje. Okrajově uvádí algoritmické myšlení ve vzdělávacích oblastech Matematika a její aplikace a Informační a komunikační technologie. V oblasti ICT se zaměřuje především na znalosti a uživatelské dovednosti žáků, což je s ohledem na rychlost změn v této oblasti zcela nedostačující. Nebere tak v potaz vznik nových profesí, v nichž je způsobilost pracovat s digitálními technologiemi zcela zásadní.

Potřeba rozvíjet kompetence v oblasti informačních technologií, především schopnost řešení problémů, je pro budoucnost našich žáků zcela zásadní. Výsledky pedagogického experimentu dokazují, že výuka programování ve Scratch může významným dílem přispět k naplnění těchto cílů, a proto by bylo vhodné výuku ve Scratch do RVP ZV zařadit. Tomu však současná hodinová dotace vzdělávací oblasti Informační a komunikační technologie nevyhovuje.

Jednou z možností je zásadní revize RVP ZV a transformace obsahu vzdělávání jednotlivých vzdělávacích oblastí. V oblasti ICT je zapotřebí zaměřit se více na témata z informatiky, na rozvoj informatického myšlení stejně, jak se tomu v současnosti děje i v zahraničí. Rovněž by bylo vhodné začít rozvíjet některé koncepty už v nižších ročnících 1. stupně ZŠ v aktivitách bez použití počítače jako součást vzdělávacího cíle rozvíjet logické, algoritmické a informatické myšlení.

Po zkušenostech získaných díky tomuto pedagogickému experimentu lze konstatovat, že výuka algoritmizace pomocí osvojování základů programování v prostředí Scratch má své opodstatnění a je tedy vhodné ji do vzdělávacího obsahu RVP ZV zařadit.

ZÁVĚR

Digitální technologie se staly naprosto běžnou součástí našeho života. Nejsou již pouze výsadou skupiny expertů a nástrojem pro odbornou práci, ale plně se integrovaly do prakticky všech oborů lidské činnosti a staly se nezbytnou součástí studia již od základní školy. Pro řadu lidí je dnes například psaní na počítači, ať již v textových editorech či v případě komunikace, samozřejmé a mnohem častější než klasické psaní na papír. Přesto práce s klávesnicí stále ještě není součástí výuky psaní. Přitom dovednosti psaní, úpravy textu, osvojení typografických pravidel korespondují s cíli výuky českého jazyka. Zpracování dat, grafy, výpočty v tabulkových editorech patří do matematiky, nahrávání a editace zvuku do hudební výchovy, práce s videem a grafikou do výchovy výtvarné apod. Základní uživatelské dovednosti při práci s výpočetní technikou, včetně práce s konkrétními programy, by proto měly být v rámci RVP převedeny do vzdělávacích oblastí, které se danou problematikou zabývají, a informaticky zaměřené předměty by se mohly zaměřit na rozvoj informatického myšlení a klíčové koncepty a dovednosti z informatiky. Je třeba vybavit žáky nejenom znalostmi, ale především schopnostmi a dovednostmi, které jim umožní v současném světě uspět. A bez schopnosti logického, kritického a algoritmického myšlení toho lze dosáhnout jen velmi těžko.

Digitální technologie se stále vyvíjejí a jejich tvůrci se stále více snaží přiblížit je člověku. Využíváme techniku, která za nás pracuje s účinností, které sami nejsme schopni. Lidé začali spoléhat na počítače, aniž by chápali, jak vlastně fungují. Proto je třeba rozvíjet informatické, a tedy i algoritmické myšlení.

V rámci podmínek pedagogického experimentu se prokázalo, že programování a algoritmické aktivity mohou být jedním ze způsobů, jak myšlení žáků rozvíjet. Přestože vzorek žáků zapojených do pedagogického experimentu nebyl velký a reprezentativní a zkušenosti tudíž nelze zobecnit, výsledky ukazují na pozitivní posun ve schopnostech většiny žáků řešit problémové úlohy algoritmické povahy oproti době před započetím experimentu. Lze tedy konstatovat, že výuka algoritmizace pomocí osvojování základů programování v prostředí Scratch má své opodstatnění a je tedy vhodné ji do vzdělávacího obsahu RVP ZV zařadit.

V současnosti se však většina žáků s programováním může setkat až na středních školách, což je pozdě. Je třeba začít již mnohem dříve na základní škole. Scratch je jedním z prostředí, v němž lze nenásilně pomocí tvořivé a zábavné činnosti rozvíjet schopnost algoritmizace. Naprostá většina profesionálních programovacích jazyků je totiž pro žáky základních škol v tomto věku nevhodná. Důležité je zaměřit se ve vzdělávání na rozvoj obecných schopností hledat strategie a postupy pro řešení problémů. Dovednosti pohlížet na problémy podobně jak to dělají počítače, mohou žáci využít nejen při práci s výpočetní technikou, ale i v každodenním běžném životě.

V průběhu výuky se osvědčilo kombinovat aktivity bez počítače s aktivitami ve Scratch. Rovněž se ukázalo, že je velice důležité, aby učitel okamžitě reagoval na konkrétní problémy žáků při práci se Scratch, pomohl jim dalšími úkoly a otázkami k pochopení konceptů a nástrojů, se kterými žáci ve Scratch pracovali.

Cílem zavedení výuky algoritmického myšlení do školního vzdělávání by neměla být výchova budoucích programátorů, ale snaha o rozšíření a prohloubení kompetencí žáků řešit problémy a přispět k rozvoji jejich myšlení. Porozumění otázkám automatizace procesů s využitím počítačových technologií přispěje k hledání efektivnějších postupů, napomůže technickému vývoji a ušetří lidem čas a lidskou práci. Jestliže dosud čtení, psaní a počítání patřilo k základním gramotnostem, pak v současné společnosti již k těmto gramotnostem náleží i dovednost informatického myšlení a je zapotřebí tomu přizpůsobit i vzdělávací obsah.

Studie ze zahraničí ukazují, že výuka algoritmizace na základní školu patří a náš pedagogický experiment to potvrdil. Výuka algoritmického myšlení může začít již u malých dětí, musí být však přiměřená jejich mentálnímu vývoji. V průběhu experimentu vyvstala celá řada problémů, které přímo souvisely s věkem žáků a s mezipředmětovými vazbami, například neznalost souřadnicového systému, záporných čísel, proměnných či pojmu úhel. Proto je mimořádně důležité zaměřit pozornost na mezipředmětové souvislosti a výuku jim přizpůsobit.

Algoritmizace byla především pro žáky 5. ročníku naprostou novinkou, a proto se s ní v průběhu výuky někteří žáci potýkali. Mezi žáky 6. ročníku se zase vyskytovali jedinci, kterým se probírané učivo jevilo příliš snadné a vhodné spíše pro mladší žáky. Výsledky

závěrečného dotazníku však ukazují, že práce na závěrečných projektech žáky nadchla. Potvrdilo se tedy, že výuka základů programování prostřednictvím tvorby her je velmi vhodná, neboť vidina naprogramování vlastní počítačové hry byla pro žáky velmi silnou motivací pro další práci a učení.

Výuka algoritmizace a základů programování není v rozsahu uskutečněném v rámci pedagogického experimentu pro svou časovou náročnost vhodná k zařazení do běžné výuky. Bylo by vhodnější v každém ročníku po celou dobu výuky pracovat na jednom komplexnějším projektu, rozděleném na menší, dílčí úlohy, ve kterých by se žáci průběžně učili jednotlivým programovacím konceptům. Osvojené dovednosti při práci ve Scratch by tak žáci ihned využívali pro zakomponování do svých projektů, například počítačových her, přičemž by se časová náročnost výuky snížila.

Pro samotnou výuku základů programování je třeba zvolit vhodné výukové (nejlépe vizuální) prostředí, ve kterém si mohou žáci sami své návrhy postupů vyzkoušet. Výuka programování pak povede k rozvoji algoritmického myšlení a úspěšné řešení problémových úloh bude motivovat žáky k dalšímu učení.

Ukázalo se, že Scratch je právě takovým nástrojem, který splňuje tyto požadavky. Jeho využití podporuje konstruktivní metody výuky. Scratch žákům nenásilnou formou pomáhal pochopit základní programovací prvky a principy. Vytvořené a sdílené programy byly pro žáky inspirací a možnost vyprodukovat vlastními silami program, který vykonává to, co sami zamýšleli, byl pro ně silnou motivací.

Prostředí Scratch je velmi vhodné pro první kroky ve výuce programování žáků 5. a 6. tříd především díky vizuálnímu prostředí a uživatelské přívětivosti, neboť jeho pracovní prostředí je až na výjimky dostatečně přehledné a má vlastnosti, které podporují žáky v aktivním experimentování. Scratch nabízí prostor pro kreativitu žáků a práci a učení vlastním tempem, poskytuje žákům okamžitou zpětnou vazbu a umožňuje vzájemnou spolupráci dětí. To jsou důvody, proč je pro řadu žáků činnost v prostředí Scratch spíše než učením, zábavou.

SEZNAM ZDROJŮ

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). *A K-6 Computational Thinking Curriculum Framework- Implications for Teacher Knowledge. Educational Technology & Society*, 19(3), 47–57.

Armoni, M. (2010). *On Teaching Topics in Computer Science Theory*. ACM Inroads, 1(1), 21–22.

Barr, V., & Stephenson, C. (2011). *Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?* ACM Inroads, 2(1), 48–54.

Bobřík informatiky: Informatická soutěž pro žáky základních a středních škol. iBobr [online]. KIN PF JČU, 2018 [cit. 2018-01-13]. Dostupné z: https://www.ibobr.cz/

Bocconi, S., A. Chioccariello, G. Dettori, A. Ferrari a K. Ngelhardt. (2016). *Developing Computational Thinking in Compulsory Education: Implications for policy and practice* [online]. Luxembourg: Publications Office of the European Union, 2016 [cit. 2018-03-27]. Dostupné z: http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf. ISBN 978-92-79-64442-9

Creative Learning Exchange. (2015). *Using System Dynamics and Systems Thinking* (SD/ST) Tools and Learning Strategies to Build Science, Technology, Engineering, and Math Excellence.

CSTA & ISTE. (2009). *Computational thinking across the Curriculum.* Retrieved from http://csta.acm.org/Curriculum/sub/CurrFiles/CTExamplesTable.pdf

Černochová, M., Šandová, H. (2015). *První ohlédnutí za výukou základů programování ve Scratch na ZŠ aneb Čím nás žáci překvapili i zaskočili, co musíme příště dělat jinak.* In: DidInfo 2015. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2015, s. 7-11. ISBN 978-80-557-0852-2.

Dohnal, P. (2008). *Programování a programovací jazyky ve vzdělávání na ZŠ* [online]. Brno, 2009 [cit. 2018-04-15]. Dostupné z: http://is.muni.cz/th/72886/pedf_m/.

Fojtík, R. (2017). *Výuka programování pomocí robota Ozobot.* In: DidInfo&DidactIG 2017. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2017, s. 54-58. ISBN 978-80-557-1216-1. ISSN 2454-051X.

Futschek, G. (2006). *Algorithmic Thinking: The Key for Understanding Computer Science.* In Lecture Notes in Computer Science 4226, Springer, pp. 159–168.

Futschek, G., Moschitz, J. (2010). *Developing Algorithmic Thinking by Inventing and Playing Algorithms* [online]. 2010, Constructionism 2010, Paris [cit. 2018-02-02]. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_187461.pdf

Gretter, S., & Yadav, A. (2016). *Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills.* TechTrends, s. 1–7.

Grover, S., & Pea, R. (2013). *Computational Thinking in K–12 A Review of the State of the Field. Educational Researcher*, 42(1), 38–43.

Havelková, H. (2008). *Algoritmy* [online]. Poslední aktualizace 2008-09-30, [cit. 2018-03-11]. Dostupné z: http://wvc.pf.jcu.cz/ki/data/files/93prg1.ppt

Hájek, Z., Vasaráb, P. (2015). *Skúsenosti s programovaním na základnej škole* – In: DidInfo 2015. Banská Bystrica : Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2015. s. 72-77. ISBN 978-80-557-0852-2.

Charlton, P., & Luckin, R. (2012). *Time to re-load? Computational Thinking and Computer Science in Schools* (Briefing 2, 27 April). The London Knowledge Lab.

Kalaš, I. (2017). *ScratchMaths: vzdelávací obsah a princípy tvorby.* In: DidInfo&DidactIG 2017. Banska Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2017, s. 16-24. ISBN 978-80-557-1216-1. ISSN 2454-051X.

Kanáliková, A. (2015). *Programovanie, alebo hra?*. In: DidInfo 2015. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2015, s. 89-92. ISBN 978-80-557-0852-2.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011). *Computational thinking for youth in practice.* ACM Inroads, 2(1), 32–37.

Lessner, D. (2015). *How Do We Translate Computional Tjinking Into Czech?: Jak si přeložíme „Computional thinking?“* [online]. 2015, 10 [cit. 2018-01-29]. Dostupné z: http://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf

Lifelong Kindergarten Group, MIT Media Lab. (2017). *Programming Concepts and Skills Supported in Scratch.* In: *ScratchEd* [online]. Poslední aktualizace 2017-03-05 [cit. 2017-12-15]. Dostupné z: http://scratched.gse.harvard.edu/sites/default/files/scratchprogrammingconcepts-v14.pdf

Lifelong Kindergarten Group, MIT Media Lab. (2018). *AboutScratch.* [online]. Poslední aktualizace 2018-02-30 [cit. 2018-03-11]. Dostupné z: https://scratch.mit.edu/about/.

Majherová, J., Palásthy, H., Králík, V., Lajčiak, P. (2014) *Príprava budúcich učiteľov informatiky a nové trendy vo výučbe programovania*. In: DidInfo 2014. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2014, s. 97-102. ISBN 978-80-557-0698-6.

Majherová, J., Petrušková, H., Valuška, P. (2017) *Interaktívne učebné materiály vo vyučovaní algoritmizácie.* In: DidInfo&DidactIG 2017. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2017, s. 101-104. ISBN 978-80-557-1216-1. ISSN 2454-051X.

MŠMT. (2014). *Strategie digitálního vzdělávání do roku 2020*. In: Ministerstvo školství, mládeže a tělovýchovy [online]. 2014 [cit. 2018-04-08]. Dostupné z: http://www.msmt.cz/ministerstvo/strategie-digitalniho-vzdelavani-do-roku-2020

Nagy, M. (2017). *Rozvíjanie čitateľskej gramotnosti prostredníctvom detského programovacieho jazyka Scratch.* In: DidInfo&DidactIG 2017. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2017, s. 117-121. ISBN 978-80-557-1216-1. ISSN 2454-051X.

Pecinovský, R. (2011). *Tvorba učebnic a kurzů programování* [online]. Žilina, 7 s. [cit. 2017-12-13]. Příspěvek na konferenci Objekty 2011. Dostupné z: http://www.vyuka.pecinovsky.cz/prispevky/2011_OB_Tvorba%20u%C4%8Debnic%20programov%C3%A1n%C3%AD.pdf

Piaget, J., Inhelder, B. (2014). *Psychologie dítěte*. Přeložila: Eva Vyskočilová. Praha: Portál, 2014. Klasici. ISBN 978-80-262-0691-0.

Pitner, T. (2000). *Výuka programování na základní a střední škole.* [online]. 13-3-2000, [cit. 2017-12-15]. Dostupný z: http://www.fi.muni.cz/~tomp/semuc/text_pitner.html

RVP ZV (2017). *Rámcový vzdělávací program pro základní vzdělávání.* Praha: MŠMT, 2017, ročník 2017, 82/2015 Sb. Dostupné také z: http://www.msmt.cz/file/43792/

Selby, C. C., & Woollard, J. (2013). *Computational Thinking: The Developing Definition. University of Southampton* (E-prints).

Schubert, S., Schwill, A. (2011). *Didaktik der Informatik*. 2. Aufl. Heidelberg: Spektrum Akademischer Verl, 2011. ISBN 9783827426529.

Skalka, J. et al. (2007). *Informatika na maturity a prijímacie skúšky.* Nitra: ENIGMA, 2007. 460 s. ISBN 978-80-89132-50-8.

Vaníček, J. (2015). *Programování ve Scratch badatelsky orientovaným přístupem* – In: DidInfo 2015. Banská Bystrica : Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2015. s. 169-174. ISBN 978-80-557-0852-2.

Vaníček, J. (2016). *Výuka algoritmizace patří především do informatiky.* In: Počítač ve škole 2016 [online]. Nové Město na Moravě: Gymnázium Vincence Makovského se sportovními třídami, 2016, s. 5 [cit. 2018-02-02]. ISBN 978-80-905765-6-8. Dostupné z: https://www.pocitacveskole.cz/system/files/soubory/sbornik/2016/vanicek1.pdf

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). *Defining Computational Thinking for Mathematics and Science Classrooms.* Journal of Science Education and Technology, 1–21.

Wing, J. (2006). *Computational thinking.* In: Communications of the *ACM.* 2006, [cit. 2018-02-28] vol. 49, no. 3, pp. 33–35. ISSN 00010782.

Wing, J. (2010). *Computational Thinking: What and Why?* [online]. 2010 [cit. 2018-02-28]. Dostupné z: https://www.cs.cmu.edu/~CompThink/papers/Th eLinkWing.pdf

SEZNAM ZKRATEK

CSTA	Computer Science Teachers Association (Asociace učitelů informatiky)
ICT	Information and communications technology (Informační a komunikační technologie)
ISTE	International Society for Technology in Education (Mezinárodní společnost pro technologie ve vzdělávání)
IT	Information technology (informační technologie)
PC	Personal computer (osobní počítač)
RVP	Rámcově vzdělávací plán
ŠVP	Školní vzdělávací program
ZV	Základní vzdělávání

SEZNAM OBRÁZKŮ, GRAFŮ A TABULEK

SEZNAM PŘÍLOH

Příloha A – DOT 01 – Dotazník vztah žáků k programování

žákovský dotazník č. ☐

Pohlaví dívka ☐ chlapec ☐ **Třída** …….. **Věk** ……..

1. Kolik dní v týdnu používáš výpočetní techniku – počítač, tablet, notebook nebo smartphone (mimo telefonování)? Zaškrtni počet dnů.

0	1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐	☐

2. Kolik času průměrně denně věnuješ práci na počítači? (Notebooku, tabletu, smartphonu apod.)

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

3. Kolik času průměrně denně věnuješ hraní počítačových her?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

4. Kolik času průměrně denně věnuješ komunikaci pomocí sociálních sítí?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

5. Kolik času průměrně denně při práci na počítači věnuješ přípravě do školy?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

6. Kolik času průměrně denně věnuješ relaxaci – brouzdání po internetu, sledování videa a podobně?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

7. Kolik času průměrně denně při práci na počítači věnuješ tvořivé činnosti – kreslení, tvůrčímu psaní, úpravě fotografií, hudby, videa a podobně?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více
☐	☐	☐	☐	☐

8. Pokud jsi nezvolil/a možnost žádný, napiš, o jaké činnosti se jedná.

1.	2.	3.

napiš případné další činnosti.....

9. Kolik času z tvé tvořivé činnosti představuje přibližně za 1 den programování?

žádný	nejvíce 1 hodinu	1-2 hodiny	3-4 hodiny	5 hodin a více (vepiš číslo)
☐	☐	☐	☐	☐

10. Jak pokročilý/á uživatel/ka výpočetní techniky jsi?

Na stupnici od 0 do 10 zaškrtni číslo, které vystihuje tvoje znalosti o počítačích.

0	1	2	3	4	5	6	7	8	9	10
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

žádné ◄——————► průměrné ◄——————► vynikající

Na stupnici od 0 do 10 zaškrtni číslo, které vystihuje tvoje počítačové **dovednosti.** Kolik toho umíš?

0	1	2	3	4	5	6	7	8	9	10
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

žádné ◄——————► průměrné ◄——————► vynikající

Kontrolují rodiče, kolik času věnuješ počítači?

☐ Ano ☐ Ne

Kontrolují rodiče, co děláš na počítači?

☐ Ano ☐ Ne

Využívají tvoji rodiče ve svém povolání počítač?

☐ Ano ☐ Ne

Věnuje se někdo z tvé rodiny programování?

☐ Ano ☐ Ne

Máš doma programovatelné hračky?

☐ Ano ☐ Ne

Programoval/a jsi už někdy v minulosti?

☐ Ano ☐ Ne

17. Pokud ano, napiš, v čem jsi programoval/a

……

18. Napiš pět slov, která tě napadnou v souvislosti se slovem počítačový program.

1.	2.	3.	4.	5.

19. Napiš pět zařízení, se kterými se denně setkáváš, a která potřebují pro své fungování počítačový program.

1.	2.	3.	4.	5.

20. Napiš pět počítačových programů, které znáš.

1.	2.	3.	4.	5.

21. Napiš pět slov, která tě napadnou v souvislosti se slovem počítačová hra.

1.	2.	3.	4.	5.

22. Napiš, jak asi vznikají počítačové programy.

.....

.....

.....

23. Napiš, z čeho se skládá počítačový program. (Jak asi vypadá uvnitř?)

.....

.....

.....

24. Napiš, jak nejčastěji získáváš software – počítačové programy (hry, grafické editory apod.)

1.	2.	3.

Příloha B – PRACOVNÍ LIST č. 1 – Algoritmus

Napište návod – postup práce. Například přípravu vaření čaje (míchaná vejce, palačinky, výměna žárovky apod.)

1. .....
2. .....
3. .....
4. .....
5. .....
6. .....
7. .....
8. .....
9. .....
10. .....

Je váš postup jednoznačný?

ano ☐ ne ☐

Dal by se váš postup opakovat?

ano ☐ ne ☐

Dal by se použít pro víc situací?

ano ☐ ne ☐

Má váš postup konec?

ano ☐ ne ☐

Napište alespoň tři běžné životní situace, ve kterých používáte postup, který lze opakovat. Mohli bychom říci, že se jedná o algoritmus?

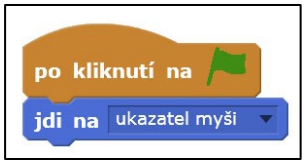
1. .....
2. .....
3. .....

Příloha C – TEST B1 – Porozumění konceptům ve Scratch

Otázka 1 - Co vykoná tento kód?

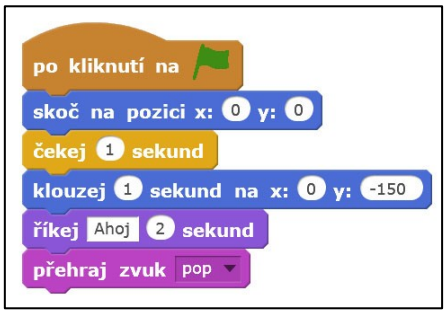
a) Po stisknutí klávesy „k“ posune objekt o 10 bodů vpravo
b) Po stisknutí klávesy „k“ posune objekt o 10 bodů vlevo
c) Po stisknutí „klávesa“ posune objekt o 10 bodů dolů
d) Po stisknutí klávesy „k“ posune objekt o 10 kroků dopředu

Otázka 2 - Co vykoná tento kód?



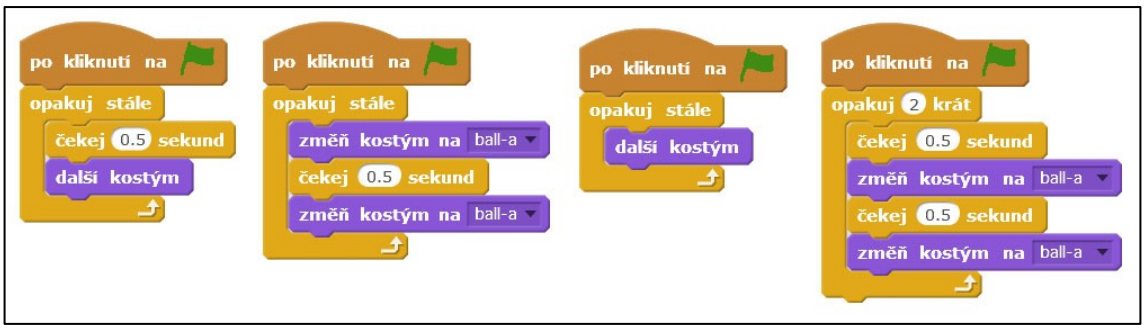
a) Po startu programu přesune kurzor myši na X=0; Y=0
b) Po stisknutí klávesy „praporek“ posune objekt pomalu k myši
c) Po startu programu se objekt bude stále držet na kurzoru myši
d) Po startu programu objekt jedenkrát skočí na kurzor myši

Otázka 3 - Co vykoná tento kód?



a) Po startu programu objekt skočí na střed plochy, bude se posunovat směrem dolů, ukáže bublinu s nápisem Ahoj a přehraje zvukový soubor s názvem „pop“
b) Po startu programu objekt skočí na souřadnice x=0 a Y=0, počká jednu sekundu, bude se posunovat směrem šikmo vpravo nahoru, přehraje zvukový soubor s názvem „pop“ a potom ukáže bublinu s nápisem Ahoj
c) Po startu programu objekt skočí na střed plochy, počká jednu sekundu, bude se posunovat směrem dolů, ukáže bublinu s nápisem Ahoj a přehraje zvukový soubor s názvem „pop“
d) Po startu programu objekt skočí na levý horní okraj plochy, počká jednu sekundu, bude se posunovat směrem dolů, přehraje zvukový soubor s názvem „Ahoj“ a přehraje zvukový soubor s názvem „pop“

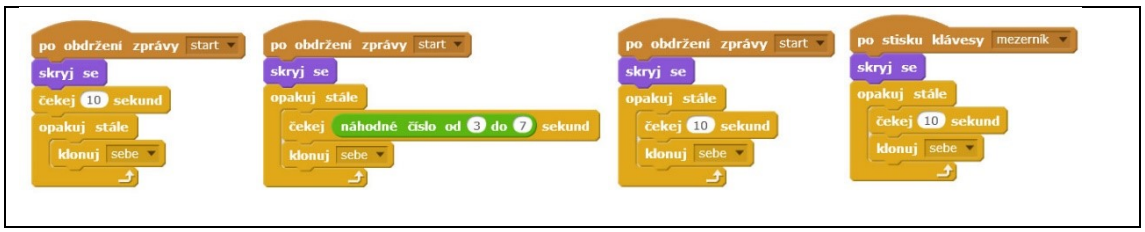
Otázka 4 – Vyber kód, který bude střídat kostýmy objektu stále dokola, vždy po půl sekundě.



Otázka 5 - Co vykoná tento kód?

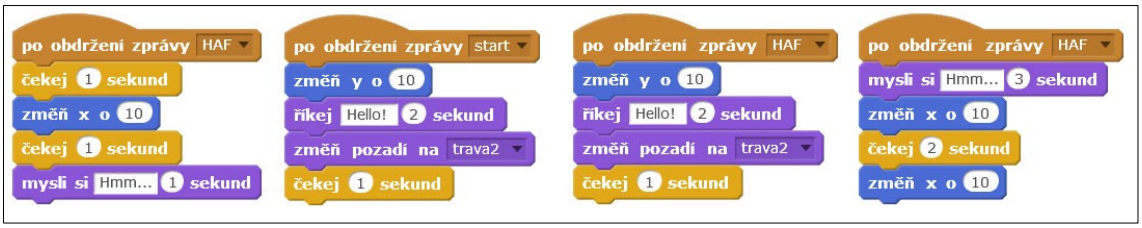
a) Po startu programu se objekt přesune 10 kroků dopředu, když během toho narazí na okraj, odrazí se
b) Po startu programu se objekt bude pohybovat pouze vodorovně po ploše programu, když narazí na okraj, odrazí se
c) Po startu programu se objekt bude pohybovat po ploše programu, když narazí na okraj, odrazí se (směr neznáme, nevíme kam je objekt natočen)
d) Po startu programu se objekt přesune o 10 kroků dopředu, odrazí se, posune se o 10 kroků, odrazí se a tak stále dokola

Otázka 6 - Z následujících čtyř kódů vyber ten, který po obdržení zprávy start objekt nejprve skryje a každých 10 sekund vytvoří jeho klon. Označ ho křížkem.



Otázka 7 – Který z následujících kódů po startu posune objekt míče ze středu scény ve směru šipky doleva dolů? Označ správnou možnost křížkem.

Otázka 8 – Z následujících čtyř kódů vyber dva, které začnou a skončí ve stejný okamžik.



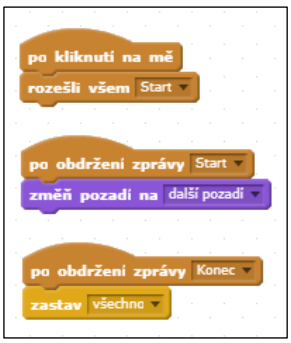
Otázka 9 - Z následujících čtyř kódů vyber dva, které pokaždé když stiskneš klávesu „a“ - budou objekt posouvat doleva tak dlouho, dokud klávesu „a“ nepustíš. Označ oba křížkem.

Otázky 10 a 11 – Jakou hodnotu budou mít proměnná časomíra a BODY po skončení těchto programů?

a) Nula
b) Jedna
c) Dva
d) Čtyři
e) Pět

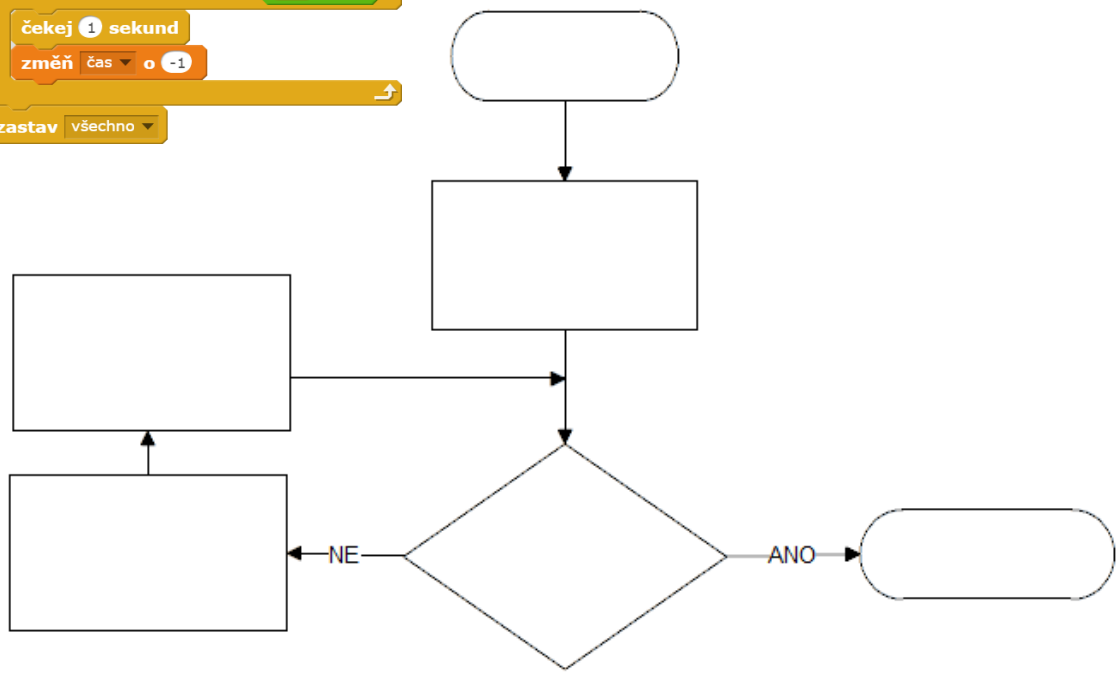
a) Nula
b) Jeden
c) Určitě méně než 10
d) Určitě více než 10
e) Nelze určit

Otázka 12 - Vyber variantu odpovědi, která odpovídá tomuto programu (program neobsahuje žádný další kód):

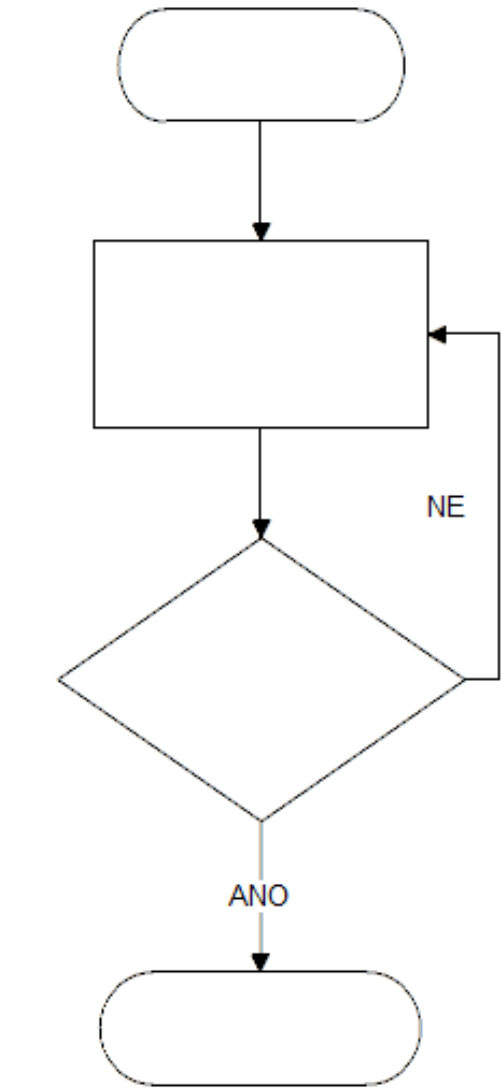


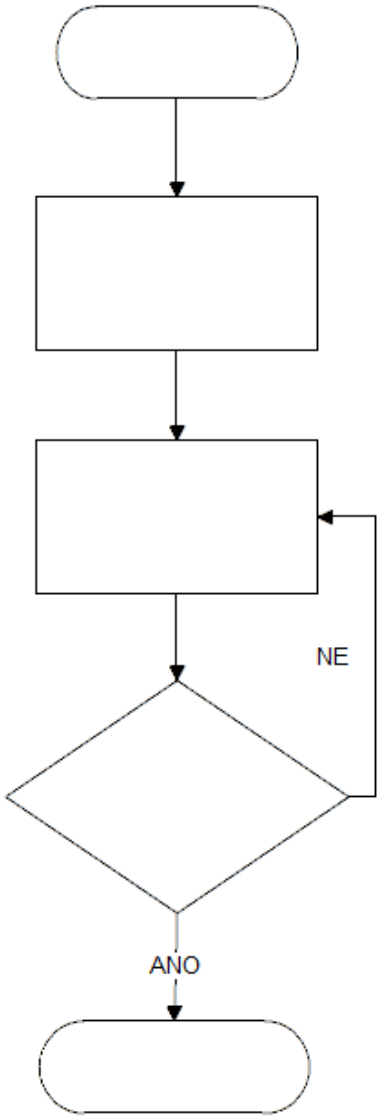
a) Po kliknutí na postavu (sprite), u kterého je tento kód, se změní pozadí na další pozadí.
b) Po kliknutí na zelený praporek se změní pozadí na další pozadí.
c) Po stisku jakékoli klávesy se změní pozadí na další pozadí.
d) Po kliknutí na postavu (sprite), u kterého je tento kód, se změní pozadí na další pozadí a program bude ihned ukončen.

Otázka 13 – Doplň algoritmus podle tohoto scriptu.

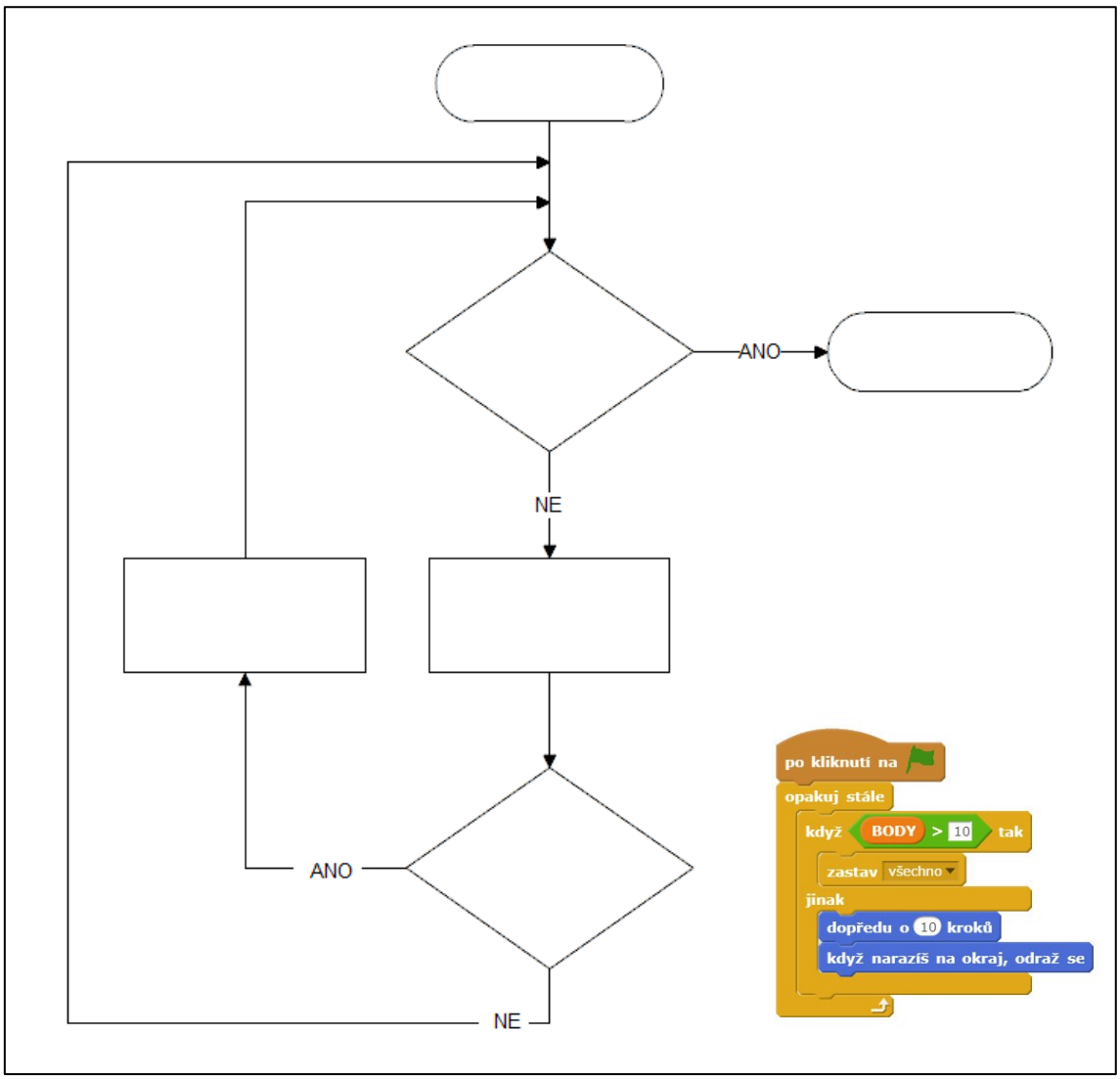


Otázky 14 a 15 - Doplň tyto algoritmy tak, aby odpovídaly scriptům ve Scratch.

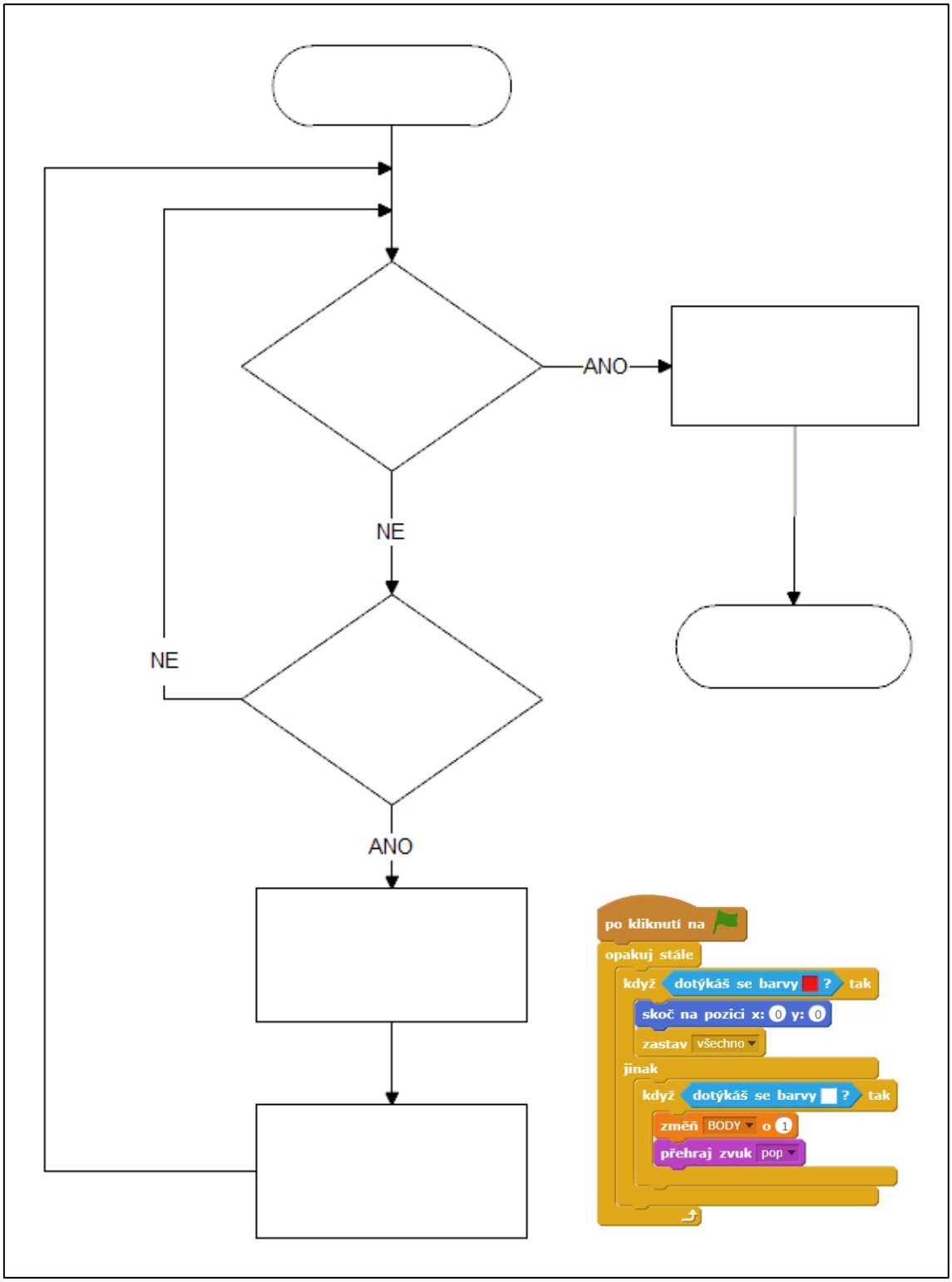




Otázka 16 - Doplň tento algoritmus tak, aby odpovídal scriptu ve Scratch.



Otázka 17 - Doplň tento algoritmus tak, aby odpovídal scriptu ve Scratch.

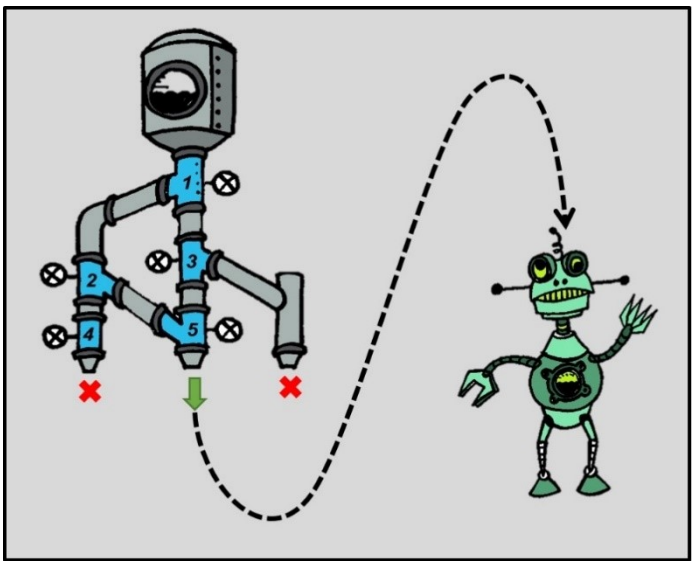


Příloha D – Návrh testových úloh – TEST A1

Úloha č. 2 – Tankování

Obtížnost: lehká **Předpokládaný čas:** 1:30 min

Robot EMIL potřebuje doplnit olej. Poraď mu, které kohouty je potřeba otevřít a které zavřít, aby dostal co nejvíce oleje z prostřední trysky. Zakroužkuj správnou volbu u všech kohoutů.



Odpověď:

Kohout 1 – otevřít – zavřít

Kohout 2 – otevřít – zavřít

Kohout 3 – otevřít – zavřít

Kohout 4 – otevřít – zavřít

Kohout 5 – otevřít – zavřít

Správná odpověď

Kohout 1 – otevřít, kohout 2 – otevřít, kohout 3 – zavřít, kohout 4 – zavřít,

kohout 5 – otevřít

Strategie řešení

Známe konec algoritmu, olej musí téci z trysky č. 5. Je dobré nejprve postupovat od konce algoritmu a určit kohouty, které musí být otevřené, aby olej vytékal z druhé trysky. Poté určit kohouty, které musí být zavřené, aby olej nevytékal tryskami po stranách.

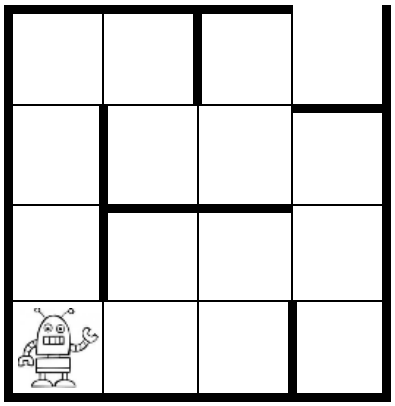
Proč?

Jedná se o algoritmickou úlohu. Na základě vstupních dat a výsledku algoritmu hledají žáci kroky mezi nimi a zkoumají, který z postupů vedl k určenému cíli.

Úloha č. 3 – ÁDA v bludišti

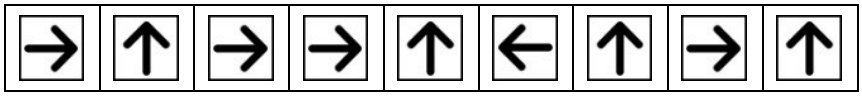
Obtížnost: lehká **Předpokládaný čas:** 1:45 min

Robot ÁDA se ztratil v bludišti, protože se vyskytly chyby v jeho programu. Doplň jeho program pomocí šipek tak, aby se po jeho spuštění dostal ven.



Odpověď:

Správná odpověď:



Strategie řešení

Najít všechna možná řešení a jejich zápis porovnat s kroky 3, 6 a 9.

Hledat stavy jednotlivých kroků a porovnávat s příslušnými kroky v programu. Známe třetí krok programu. Musíme tedy nalézt všechny pozice po kroku druhém a určit, zda z těchto míst lze jít doprava.

Grafické řešení. Zakreslení šipek symbolizujících pohyb robota přímo do bludiště umožní jejich porovnání s určenými kroky programu a následné překreslení do tabulky.

Proč?

Jedná se o algoritmickou úlohu, protože úkolem je vytvořit program pro průchod robota bludištěm. Varianty průchodu bludištěm v devíti krocích jsou tři. Je ale nutné brát v úvahu pevně určené příkazy, proto je možné vytvořit pouze jednu variantu programu.

Úloha č. 4 – Vaření čaje

Obtížnost: střední **Předpokládaný čas:** 2:30 min

Domácí robot má za úkol uvařit čaj. Má k dispozici hrnek (A) a lžičku (B), rychlovarnou konvici (C), sáček s čajem (D) a vodu (E). V jeho programu však byly vymazány některé příkazy. Doplň robotův program pomocí označení předmětů A, B, C, D a E.

Odpověď:

1. Nalij …….. do ……..
2. Zapni ……..
3. Čekej 1 minutu
4. Pokud se ..... dovařila, pokračuj na řádek 4, pokud ne, vrať se na řádek ……
5. Vlož …….. do ……..
6. Zvedni …….. a přelij …….. do ……..
7. Čekej 3 minuty
8. Vyndej …….. a přines …….. na stůl

Správná odpověď

1. Nalij E do C
2. Zapni C
3. Čekej 1 minutu
4. Pokud se E dovařila, pokračuj na řádek 4, pokud ne, vrať se na řádek 3
5. Vlož D do A
6. Zvedni C a přelij E do A
7. Čekej 3 minuty
8. Vyndej D a přines A na stůl

Strategie řešení

Doplnit popis přípravy čaje slovně a následně zaměnit slova za jejich abstraktní označení.

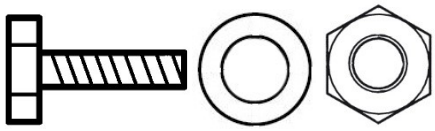
Proč?

V této úloze mají žáci za úkol vhodně doplnit algoritmus přípravy čaje. Úloha se rovněž zabývá abstrakcí informací. Žáci si musí vytvořit propojení mezi konkrétními pojmy a jejich označením.

Při řešení úloh zaměřených na popis činnosti je třeba i čtenářské gramotnosti.

Úloha č. 5 – Šroubky, podložky a matičky

Obtížnost: střední **Předpokládaný čas:** 2:30 min



Robot potřebuje najít v krabici, kde jsou smíchány různě velké šroubky, různě velké matičky a různě velké podložky, jeden šroubek, matičku a podložku, které k sobě pasují. Postupuje tak, že vždy nabere do ruky náhodně několik součástek z krabice, položí je před sebe na stůl a potom vybere největší šroubek na stole, největší matičku na stole a ostatní součástky hodí do jiné krabice. Když v první krabici nezůstane žádná součástka, robot má na stole:

a) stejně velkou matičku, šroubek a podložku
b) největší šroubek a největší matičku, které k sobě určitě pasují
c) největší šroubek a největší matičku, které k sobě mohou, ale nemusí pasovat
d) největší šroubek, největší matičku a největší podložku, které k sobě pasují
e) největší šroubek, největší matičku a největší podložku, které k sobě nemusí pasovat

Správná odpověď

a) největší šroubek a největší matičku, které k sobě nemusí pasovat

Strategie řešení

Pochopit co algoritmus vlastně dělá, není vždy zcela jasné. V tomto případě je třeba si uvědomit, že po každém kroku zůstane na stole vždy největší matička a největší šroubek. Naopak všechny podložky skončí v druhé krabici. Na konci algoritmu tedy na stole budou největší matička a šroubek z první krabice. Nikde však není dáno, že se bude jednat o stejné součástky, které k sobě pasují.

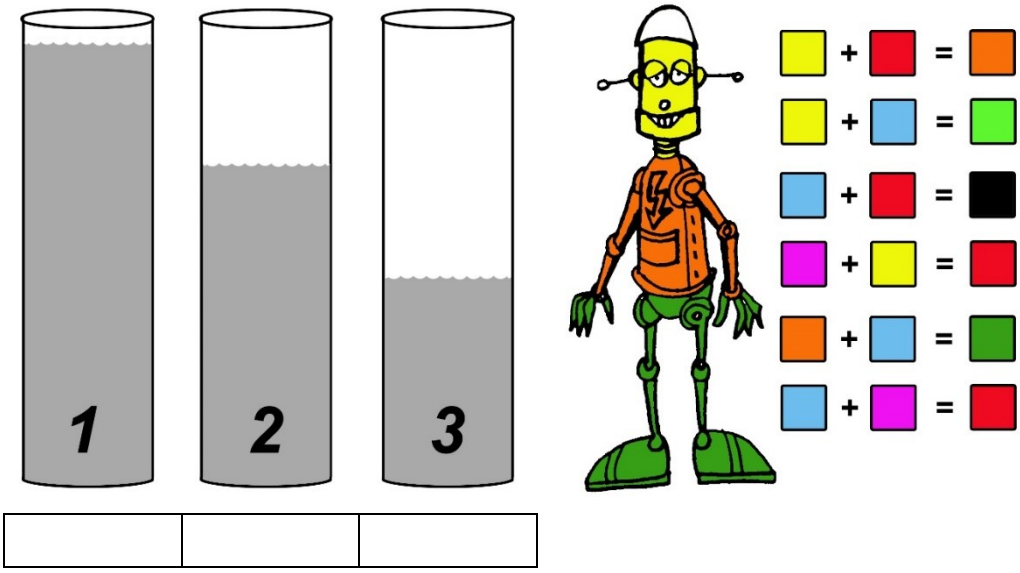
Proč?

Jedná se o úlohu, při které je třeba porozumět algoritmu, který je pevně dán. Úloha říká, jak algoritmus probíhá, ale není jasné, k jakému vede výsledku. Pro správné řešení úlohy musí žáci pochopit, že za všech okolností na stole zůstane vždy největší matička a šroubek, zatímco podložky vždy skončí v druhé krabici. Protože probereme všechny součástky, musí zůstat na stole největší matička a šroubek. Nikde ale není dáno, že největší matička a největší šroubek k sobě musí pasovat.

Úloha č. 6 – Barvení robota DUDU

Obtížnost: střední **Předpokládaný čas:** 3:45 min

Bílý robot DUDU se chtěl obarvit, proto prošel postupně třemi nádržemi s barvou. První nádrž byla téměř plná, ve druhé byla barva přibližně do dvou třetin a ve třetí do výšky asi jedné třetiny. Protože barvy nestihly na robotovi zaschnout, smíchaly se a vznikly barvy nové. Například, pokud by se nabarvil nejprve žlutou a hned potom modrou, vznikla by barva světle zelená. Jaké barvy vzniknou smícháním je vidět v tabulce.



Jaké barvy byly v nádržích? Doplň do popisků jejich názvy.

Správná odpověď: nádrž 1 = žlutá, nádrž 2 = červená, nádrž 3 = modrá

Strategie řešení

Lze postupovat od konce algoritmu, protože známe jeho výsledek a hledat všechny stavy, které k němu vedou. Nohy jsou zelené => robot prošel jinými nádržemi => barva tedy nemohla být pouze zelená. Zelená mohla vzniknout z modré a oranžové. Tělo je oranžové => třetí nádrž tedy musela být modrá.

Lze postupovat od počátku algoritmu. Čepička je bílá a hlava žlutá. Hlava prošla pouze jednou nádrží => první nádrž musela být žlutá. Žlutá + neznámá barva = oranžová => neznámá barva musí být červená…

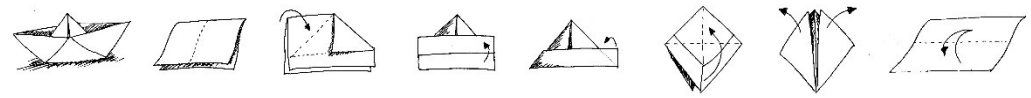
Proč?

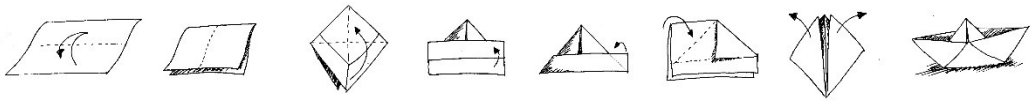
Jedná se o algoritmickou úlohu. Na základě vstupních dat a výsledku algoritmu hledají žáci kroky mezi nimi – zkoumají, který z postupů vedl k určenému cíli.

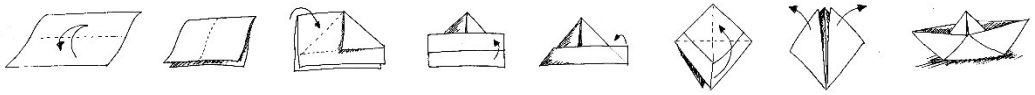
Úloha č. 7 – Lodička

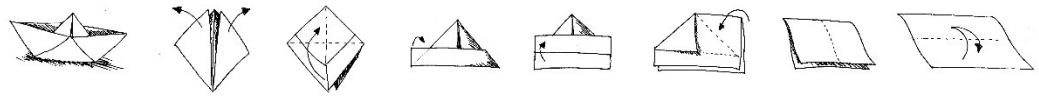
Obtížnost: střední **Předpokládaný čas:** 1:30 min

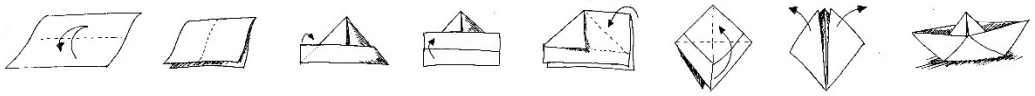
Který z následujících postupů je správným návodem na složení lodičky?

a 

b 

c 

d 

e 

Správná odpověď

Varianta c)

Chybné odpovědi

Varianta a) odpovídá rozkládání lodičky, a stejně jako d), tedy není ukončena hotovou lodičkou.
Varianta e) má prohozeny kroky 3 a 5, varianta b) má prohozeny kroky 3 a 6

Strategie řešení

Procházet jednotlivé varianty a krok za krokem hledat chyby.

Procházení kroků střídavě zepředu a zezadu zvyšuje pravděpodobnost nalezení chyb. První musí být vždy obrázek rozloženého papíru, poslední obrázek složeného parníku.

Porovnávat kroky svisle, hledat, která řada se od ostatních liší.

Proč?

Návod skládání lodičky je vlastně algoritmem – přesnou posloupností kroků k jeho výrobě. Uvědomění si správného pořadí jednotlivých kroků je potřebné k chápání funkčního programu. Jedná se o posouzení již hotového návodu, které v praxi odpovídá testování a ladění již hotového programu.

Úloha č. 8 – Výroba

Obtížnost: střední **Předpokládaný čas:** 1:30 min

Robot Z06 za 1 den vyrobí dvě své kopie. Jeho kopie vyrobí každý den rovněž dvě své kopie.

Kolik robotů Z06 bude v dílně na konci druhého dne?

Odpověď:

a) šest b) sedm c) osm d) devět e) deset

Správná odpověď:

V dílně bude 9 robotů.

První den robot 1 vyrobí roboty 2 a 3.
Druhý den roboti 1, 2 a 3 vyrobí roboty 4, 5, 6, 7, 8 a 9.
Je nutné uvažovat tak, že robot 1 pracuje i druhý den.

Strategie řešení

Grafické řešení umožní uvědomit si, že robot číslo 1 vyrábí své kopie i druhý den.

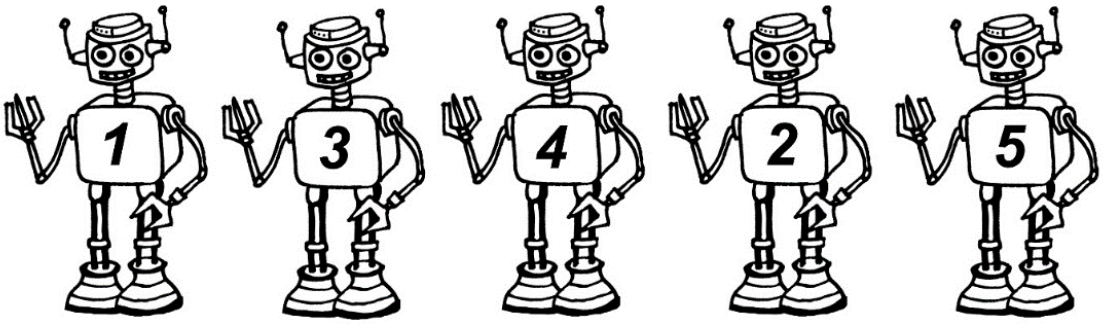
Proč?

Jedná se o úlohu zaměřenou na pochopení hierarchie klonování.

Úloha č. 9 – Řazení robotů

Varianta A – Obtížnost: lehká **Předpokládaný čas:** 1:30 min

Roboti jsou připraveni opustit továrnu. Nejsou však v pořadí, v jakém byli vyrobeni.



Roboti mají opustit továrnu v pořadí, v jakém byli vyrobeni. Mohou se ale řadit pouze tak, že si vždy dva z nich vymění svá místa. Výměna dvou robotů je považována za jeden krok. Kolik kroků je potřeba k seřazení všech robotů od prvního po pátého?

Odpověď

a) jeden b) dva c) tři d) čtyři e) pět

Správná odpověď

b) dva

Strategie řešení

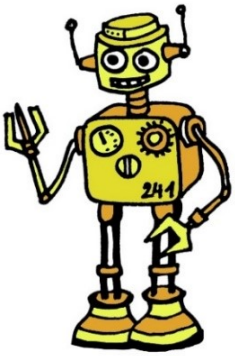
Určit roboty, kteří jsou na svých pozicích a nemusí se přesouvat. Zbývající tři roboti si mohou vyměnit místa ve dvou krocích.

Proč?

Jedná se o algoritmickou úlohu zaměřenou na řazení. Správné řazení je jedním z nejdůležitějších prvků informatického myšlení. Nalezení nejefektivnějšího postupu řazení vyžaduje algoritmické myšlení.

Úloha č. 9a a 9b – Montáž Karla

Obtížnost: střední **Předpokládaný čas:** 4:30 min

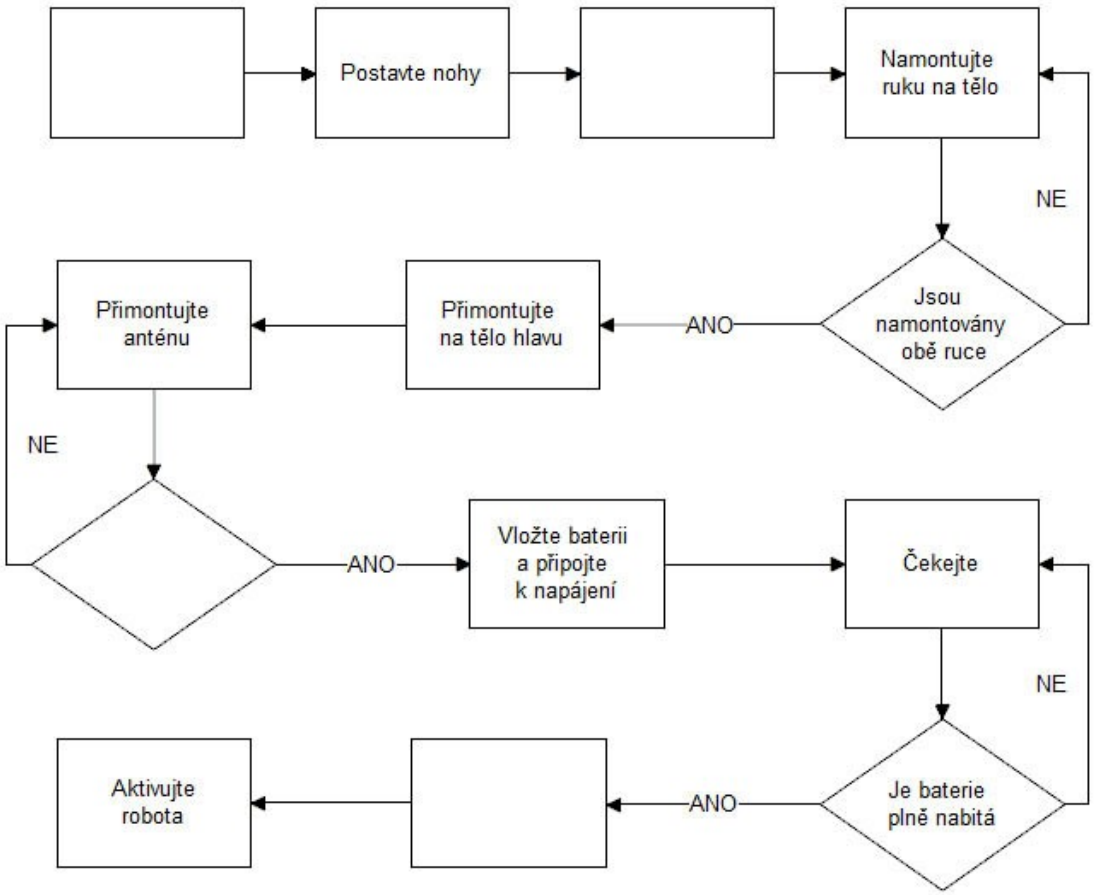


Doplň program, podle kterého bude sestaven a aktivován robot KAREL.

Do prázdných polí vepiš chybějící pokyny tak, aby program dával smysl.

Použij tyto příkazy:

- Naprogramujte robota
- Přimontujte tělo na nohy
- Vyrobte všechny součástky
- Jsou namontovány obě antény?



Stručně popište, co se v diagramu děje. Neopisujte text z rámečků.....

.....

.....

.....

.....

.....

.....

.....

Správná odpověď:

1. Vyrobte všechny součástky
2. Přimontujte tělo na nohy
3. Jsou namontovány obě antény?
4. Naprogramujte robota

Strategie řešení

Otázky jsou v kosočtvercích ⇒ doplnit nejprve jedinou otázku.

Doplnit první krok vyrobit všechny součástky.

Doplnit předposlední krok, naprogramujte robota.

Proč?

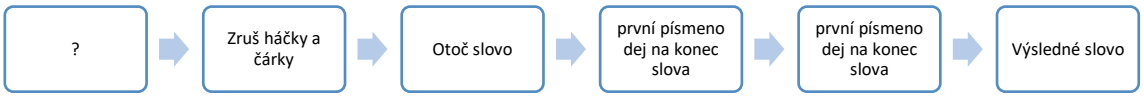
Jedná se o pochopení algoritmu, který popisuje výrobu robota. Pouze porozumění celému postupu včetně cyklů umožňuje žákům doplnit jednotlivé kroky na správná místa.

Úloha č. 10 - Program na šifrování textu

Obtížnost: lehká **Předpokládaný čas:** 2:30 min

Program na šifrování textu provádí následující kroky:

1. Otoč slovo
2. Zruš háčky a čárky
3. První písmeno dej na konec slova
4. První písmeno dej na konec slova
5. Napiš výsledné slovo



Příklad: SLON => SLON => NOLS => OLSN => LSNO

Jaké mohlo být původní slovo, které program zašifroval do shluku písmen OLSAK?

Odpověď:

a) ŠKOLA b) LÁSKO c) SLOKA d) SKÁLO e) SOKOL

Správná odpověď:

SLOKA

SLOKA => SLOKA => AKOLS => KOLSA => OLSAK

Strategie řešení

Možné odpovědi postupně měnit pomocí algoritmu. Výsledek porovnat se zadáním. Tuto variantu lze zkrátit o slovo sokol, které neobsahuje potřebná písmena.

Vytvořit algoritmus pro zpětný postup. Výsledné slovo měnit pomocí tohoto algoritmu.

Pochopit, že algoritmus má vždy podobný výsledek. Poslední dvě písmena zůstávají na posledních dvou pozicích, ale v opačném pořadí. Zbytek slova je zapsán pozpátku.

Proč?

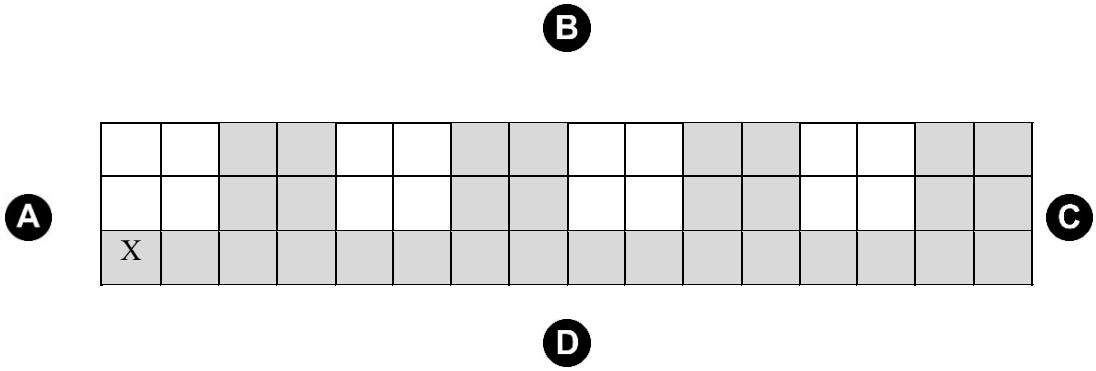
Jedná se o úlohu na pochopení funkce konkrétního algoritmu. Výsledné slovo vzniká posloupností jednotlivých kroků. Při bližším porozumění tomuto algoritmu lze dospět k jeho zjednodušení – poslední dvě písmena zůstávají na konci v opačném pořadí, zbytek slova je zapsán opačně.

Úloha č. 11 – Barvení dlaždic

Obtížnost: těžká **Předpokládaný čas:** 3:00 min

Robot má za úkol natřít šedé dlaždice modře. Nachází se na místě označeném křížkem a natře každou dlaždici, kterou přejede. Rozumí následujícím pokynům A = jede jedno pole směrem A, B = jede jedno pole směrem B, C = jede jedno pole směrem C, D = jede jedno pole směrem D. Příkazy lze zkracovat pomocí čísel. Pokud robot dostane instrukci 3B, pojede tři pole směrem B. Nikdy se nesmí vrátit zpět na čerstvě natřené dlaždice.

V jeho paměti však zůstalo jen velmi málo místa. Který nejkratší program povede k natření



šedých dlaždic na modro.

Odpověď

a) 2C 4x(2B C 2D) 2C
b) 4x (2C 2B C 2D)
c) 2x (2C 2B C 2D 2C 2B C 2D)
d) 16C 4x (2B A 2D A)
e) 8x (2C 2B C 2D)
f) 2C 2B C 2D

Správná odpověď

b) 4x (2C 2B C 2D)

Strategie řešení

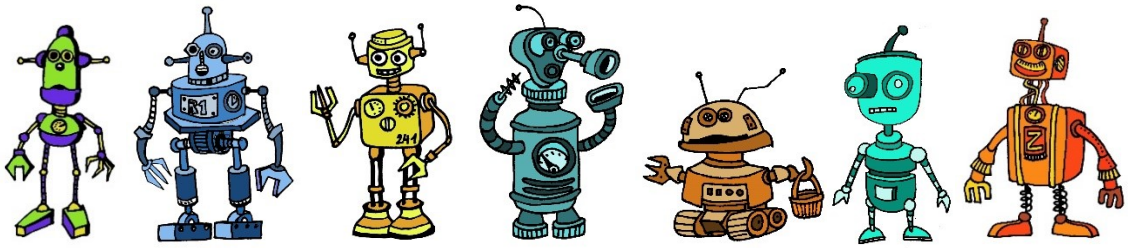
Na počátku je dobré si uvědomit, že pokud se robot nesmí vracet na již natřené dlaždice, nemůže být v jeho programu zahrnut směr A. Je důležité si také všimnout, že celý obrazec, který je třeba vybarvit, lze rozdělit na čtyři shodné části. Lze postupovat vyřazovací metodou, kdy kontrolujeme jednotlivé varianty kódu s realitou, nebo si předem vypsat správnou cestu – C C B B C D C C B B C D C C B B C D C C B B C D – zkrátit na 4x (2C 2B C 2D) a porovnat s danými možnostmi.

Proč?

Jedná se o úlohu zaměřenou na tvorbu programu. Příkazy algoritmu nutného pro obarvení konkrétních polí lze zkrátit pomocí opakování. Variant cyklu lze vytvořit více a mezi odpověďmi jsou dvě správné možnosti. Pouze jedna je však nejkratší, jak požaduje zadání úlohy.

Úloha č. 12 – Místnost plná robotů

Obtížnost: těžká **Předpokládaný čas:** 2:30 min



V místnosti je sedm robotů, kteří poslouchají tvoje příkazy. Důležitější je pro ně vždy dřívější příkaz, který se jich týká. Dostali od tebe tyto tři příkazy:

1) Kdo máte jen jednu anténu, dělejte opak dalších příkazů.
2) Kdo máte pásy, přesuňte se ke mně.
3) Kdo máte nohy, nechoďte ke mně.

Kolik robotů bude po těchto příkazech u tebe?

Odpověď:

a) žádný b) dva c) tři d) čtyři e) šest f) všichni

Správná odpověď

d) čtyři

Strategie řešení

Postupovat v algoritmu krok za krokem. Praktické je grafické řešení algoritmu. V prvním kroku označit roboty s jednou anténou, kteří budou dělat opak příkazů. V druhém označit robota s pásy, který se přesune. V třetím kroku označit roboty s jednou anténou, kteří se rovněž přesunou.

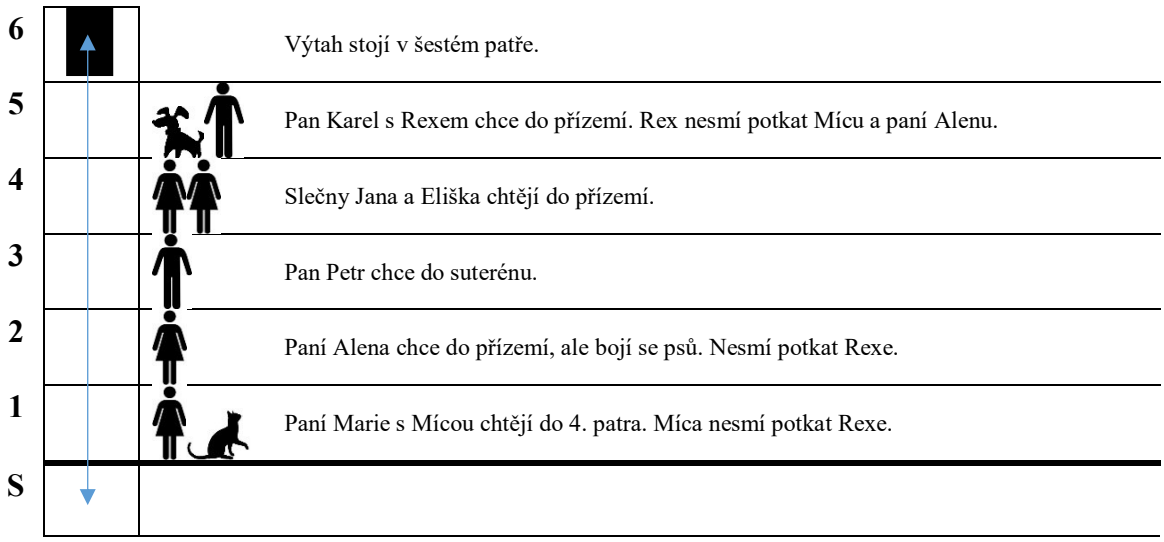
Proč?

Jedná se o úlohu na pochopení funkce konkrétního programu. Je znám algoritmus, ale nikoli jeho výsledek.

Úloha č. 13 – Výtah

Obtížnost: těžká **Předpokládaný čas:** 2:30 min

Pomocí programu ovládáš výtah, který na začátku stojí v šestém patře. Vyber program, který přepraví všechny čekající obyvatele domu tam, kam potřebují, a současně se nepotkají pes a kočka, nebo Rex a paní Alena, která se psů bojí. Do výtahu se vejdou pouze tři lidé, nebo dva lidé a jedno zvíře. Pokud výtah v patře zastaví, nastoupí všichni, kdo mohou. Suterén je označen písmenem S, přízemí číslem 1.



Odpověď:

a) 4-2-1-S-3-5-4
b) 4-2-1-4-5-3-1-S
c) 3-2-1-S-4-5-1
d) 5-1-4-3-2-1-S
e) 4-3-S-1-2-5-1

Správná odpověď

b) 4-2-1-4-5-3-1-S

Strategie řešení

Porovnávat jednotlivé kroky algoritmu se všemi podmínkami. Např. krok (4-2) jsou ve výtahu pouze 3 osoby? Potkali se paní Alena s Rexem? Potkali se Rex s Mícou? Tento způsob bude velmi zdlouhavý.

Nejprve posoudit, zda se všechny osoby dostanou na určená místa. Zkontrolovat, zda číslo patra kam chtějí, je za číslem patra, kde osoba stojí. „5 chce do 1“ – je jednička za pětkou? V případě varianty a) nikoli, proto tato varianta není správná.

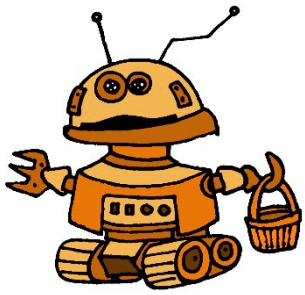
Zkontrolovat, zda čísla pater osob, která se nesmějí potkat, nejsou přímo za sebou. „Pětka nesmí potkat jedničku ani dvojku“. Varianty, kde tato čísla sousedí, jsou chybné.

Proč?

Jedná se o úlohu, kdy známe jednotlivé podmínky algoritmu a je třeba posoudit, zda hotový algoritmus tyto podmínky splňuje.

Úloha č. 14 – Cesta

Obtížnost: střední **Předpokládaný čas:** 2:30 min



Robot FANY dostala zadány do programu následující instrukce:

- Až si necháš opravit anténu, zajdi koupit konzervu Whiskas pro babiččinu kočku.
- Až všechno zařídíš, vrať se domů umýt nádobí.
- Až nakrmíš kočku, vyzvedni v cukrárně objednaný dort.
- Než půjdeš k babičce, kup půl kila hřebíků.
- Nejdřív si nech opravit zlomenou anténu.

V jakém pořadí FANY navštívila BABIČKU, OPRAVNU, CUKRÁRNU, ZVERIMEX, DOMOV a ŽELEZÁŘSTVÍ?

Vyber a zakroužkuj správnou odpověď.

Odpověď:

a) opravna > zverimex > železářství > babička > cukrárna > domov
b) zverimex > železářství > babička > cukrárna > domov > opravna
c) opravna > zverimex > železářství > babička > cukrárna > domov
d) domov > cukrárna > babička > železářství > zverimex > opravna
e) opravna > železářství > zverimex > babička > cukrárna > domov

Správná odpověď

a) opravna > zverimex > železářství > babička > cukrárna > domov

Strategie řešení

Ze zápisu jasně vyplývá, který krok bude první a který poslední. Po zapsání těchto kroků je určení dalších již poměrně snadné.

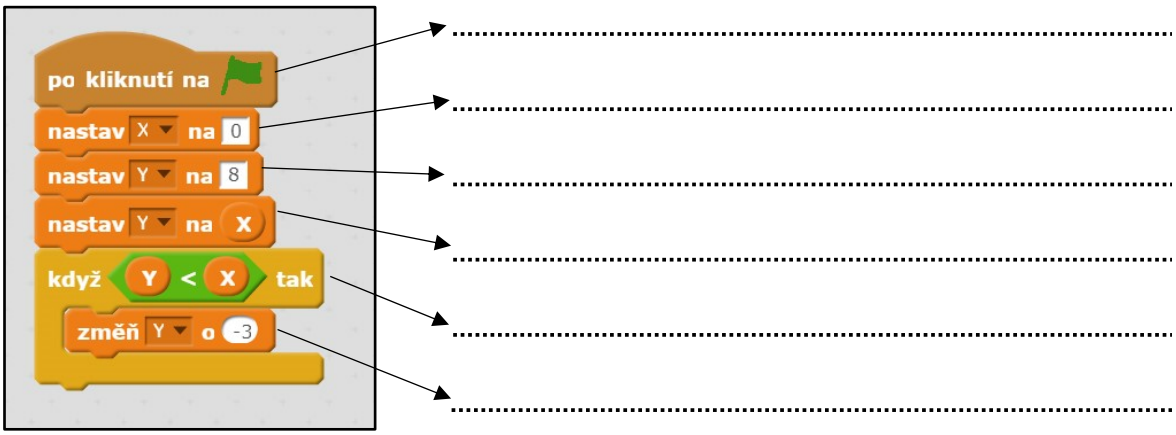
Proč?

Jedná se o algoritmickou úlohu zaměřenou na posloupnost a řazení kroků. Příkazy je třeba uspořádat podle zadaných kritérií. Žáci musí vytvořit algoritmus, který splňuje konkrétní podmínky. Pro splnění tohoto úkolu je třeba schopnost porozumění zadanému textu, tedy jistá míra čtenářské gramotnosti.

Úloha č. 15 – Dosazování

Obtížnost: těžká **Předpokládaný čas:** 2:00 min

Jakou hodnotu bude mít proměnná Y po vykonání následujícího programu? Zakroužkuj správnou odpověď. Do řádků vedle obrázku popiš svými slovy, jak programu rozumíš. Každému bodu programu odpovídá jeden řádek popisu.



Odpověď:

a) nula b) pět c) osm d) mínus tři e) jedenáct

Správná odpověď:

b) Y = 0

Strategie řešení

Postupovat v sekvenci krok za krokem a určovat, jakou hodnotu má proměnná Y. U každého řádku vypsat její hodnotu.

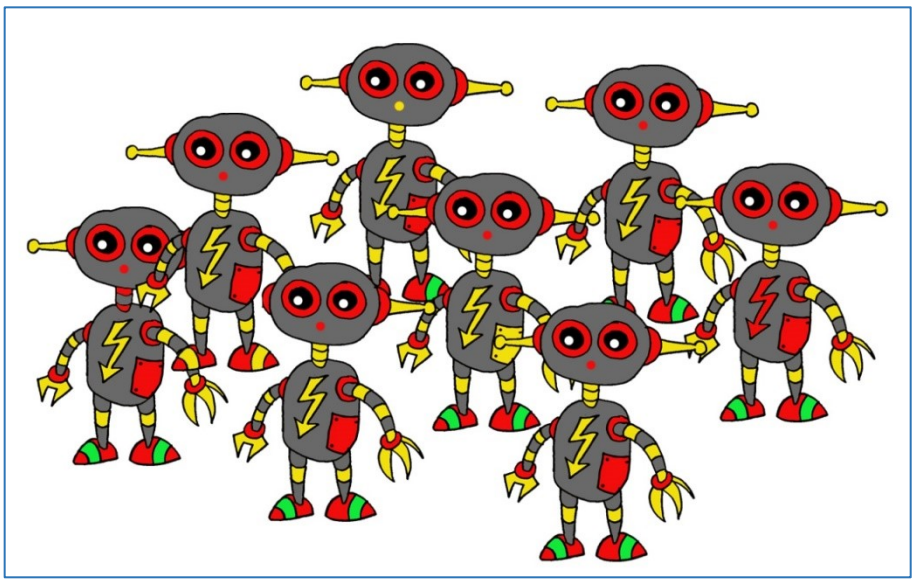
Proč?

Jedná se o úlohu na porozumění algoritmu obsahujícího proměnnou hodnotu.

Úloha č. 16 – Hra na schovávanou

Obtížnost: střední **Předpokládaný čas:** 1:30 min

Robot BE-BE-TRI se schoval mezi ostatními roboty ze stejné výrobní série. Má zelené proužky na botách, žlutý krk, dvě antény, žlutý blesk na těle, červený nos, červenou záplatu na boku a určitě si nestoupl zcela dopředu ani dozadu. Najdi robota BE-BE-TRI a označ ho.



Správná odpověď: Robot BE-BE-TRI je druhý zprava.

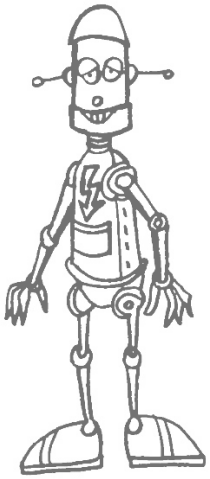
Strategie řešení

Postupovat krok za krokem a u všech kontrolovat jednu podmínku po druhé. Roboty, kteří danou podmínku nesplňují, z porovnávání vyřadit. Hledat rozdíly mezi roboty a příslušný rozdíl porovnat s podmínkami.

Proč?

Algoritmy obsahují podmínky, které programy větví. V této úloze podmínky určují, zda se může jednat o hledaného robota či nikoli. Vyloučením všech robotů, kteří všechny podmínky nesplňují, dospějeme k jedinému možnému řešení.

Příloha E – Výstupní test – TEST A2

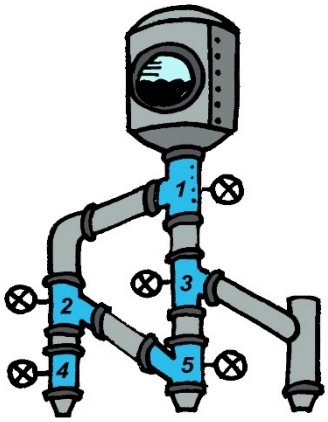


TEST A2

Dotazník číslo: ..... **třída .....**

Úloha č. 1 – Tankování

Robot EMIL potřebuje doplnit olej. Ze které trysky poteče olej, pokud jsou kohouty nastaveny podle tohoto rozpisu.



Kohout 1 – otevřeno

Kohout 2 – zavřeno

Kohout 3 – otevřeno

Kohout 4 – otevřeno

Ko| A B C

Odpověď:

a) z trysky A
b) z trysky B
c) z trysky C
d) ze žádné trysky
e) z trysek B a C

Úloha č. 2 – Vaření vajec

Domácí robot má za úkol uvařit 5 vajec natvrdo. Má k dispozici 10 vajíček (V), velkou lžíci (L), hrnec (H), elektrický vařič (EV) a minutku – měřič času (M). V jeho programu však byly vymazány některé příkazy.

Doplň robotův program pomocí označení předmětů V, L, H, M a EV.

Odpověď:

1. Připrav si pět ……
2. Nalij vodu do ……
3. Zapni ……
4. Postav …… na ……
5. Pokud se voda začala vařit, vlož do …… pět ……
6. Nastav minutku na 10 minut
7. Pokud …… zazvonila, vyndej …… z …… do misky
8. Vypni ……

Úloha č. 3 – Výroba

V prázdné dílně je v pondělí ráno pouze Robot Z0F a materiál na výrobu dalších deseti robotů. Robot Z0F do večera stihne vyrobit jednu svou identickou kopii. Přes noc nepracuje. Kolik robotů bude v dílně ve středu večer?

Odpověď:

a) šest

b) sedm

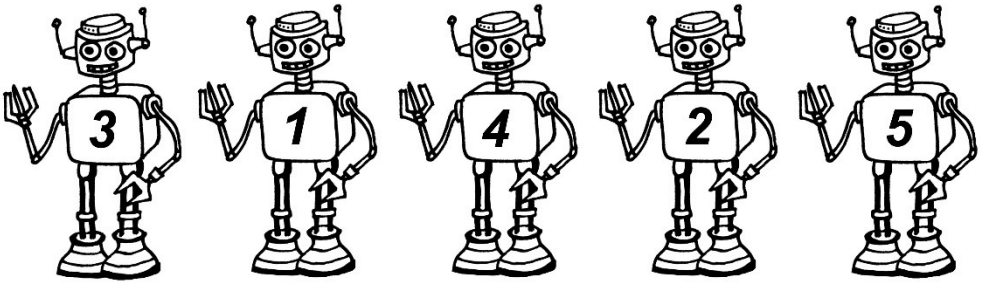
c) osm

d) devět

e) deset

Úloha č. 4 – Řazení robotů

Roboti mají opustit továrnu v pořadí, v jakém byli vyrobeni. Mohou se ale řadit pouze tak, že si vždy dva z nich vymění svá místa. Výměna dvou robotů je považována za jeden krok. Kolik kroků je potřeba k seřazení všech robotů od prvního po pátého?

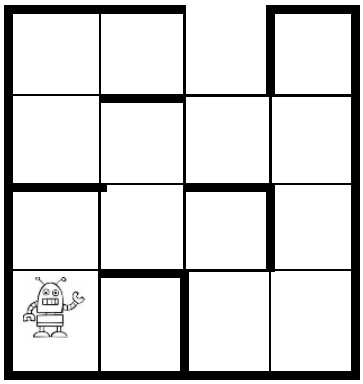


Odpověď:

a) jeden
b) dva
c) tři
d) čtyři
e) pět

Úloha č. 5 – Áda v bludišti

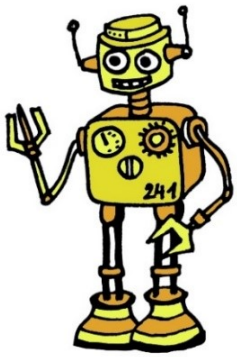
Robot ÁDA se ztratil v bludišti, protože se vyskytly chyby v jeho programu. Doplň jeho program pomocí šipek tak, aby se po jeho spuštění dostal ven.



Odpověď:

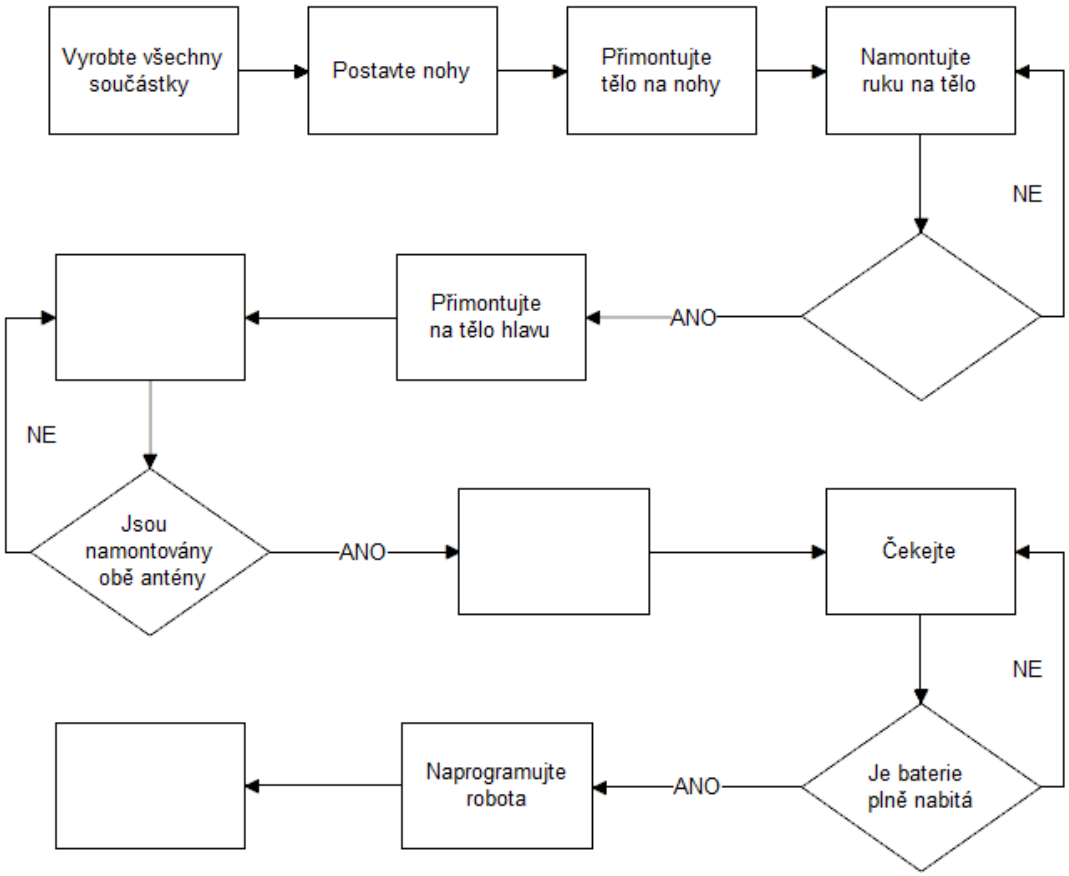
Úloha č. 6A a 6B – Montáž Karla

Doplň program, podle kterého bude sestaven a aktivován robot KAREL.



Do prázdných polí vepiš chybějící pokyny tak, aby program dával smysl. Použij tyto příkazy:

- Vložte baterii a připojte k napájení
- Přimontujte anténu
- Aktivujte robota
- Jsou namontovány obě ruce?



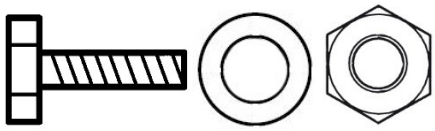
Stručně popište, co se v diagramu děje. Neopisujte text z rámečků.....

.....

.....

.....

Úloha č. 7 – Šroubky, podložky a matičky



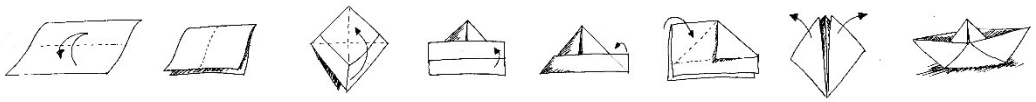
Robot potřebuje najít v krabici, kde jsou smíchány různě velké šroubky, různě velké matičky a různě velké podložky jeden šroubek, matičku a podložku, které k sobě pasují. Postupuje tak, že vždy nabere do ruky náhodně několik součástek z krabice, položí je před sebe na stůl a potom vybere největší šroubek na stole, největší matičku na stole a největší podložku na stole. Ostatní součástky hodí do jiné krabice. Když v první krabici nezůstane žádná součástka, robot má na stole:

f) stejně velkou matičku, šroubek a podložku
g) největší šroubek a největší matičku, které k sobě určitě pasují
h) největší šroubek a největší matičku, které k sobě mohou, ale nemusí pasovat
i) největší šroubek, největší matičku a největší podložku, které k sobě pasují
j) největší šroubek, největší matičku a největší podložku, které k sobě mohou, ale nemusí pasovat

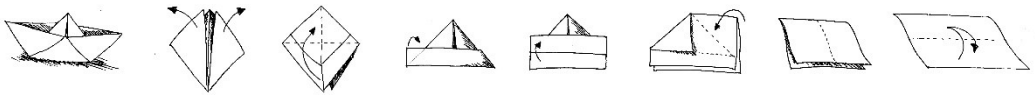
Úloha č. 8 – Lodička

Který z následujících postupů je správným návodem na složení parníku?

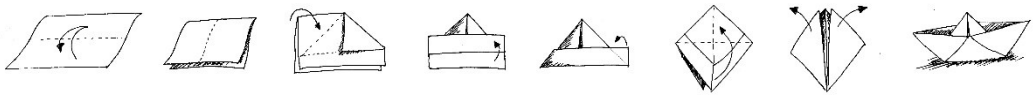
a)



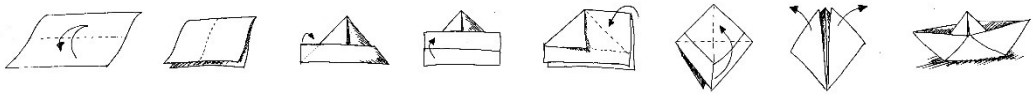
b)



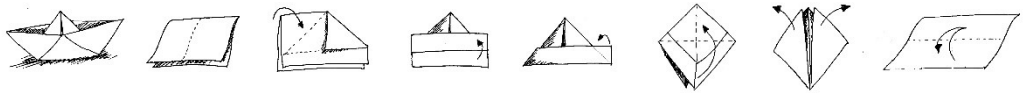
c)



d)

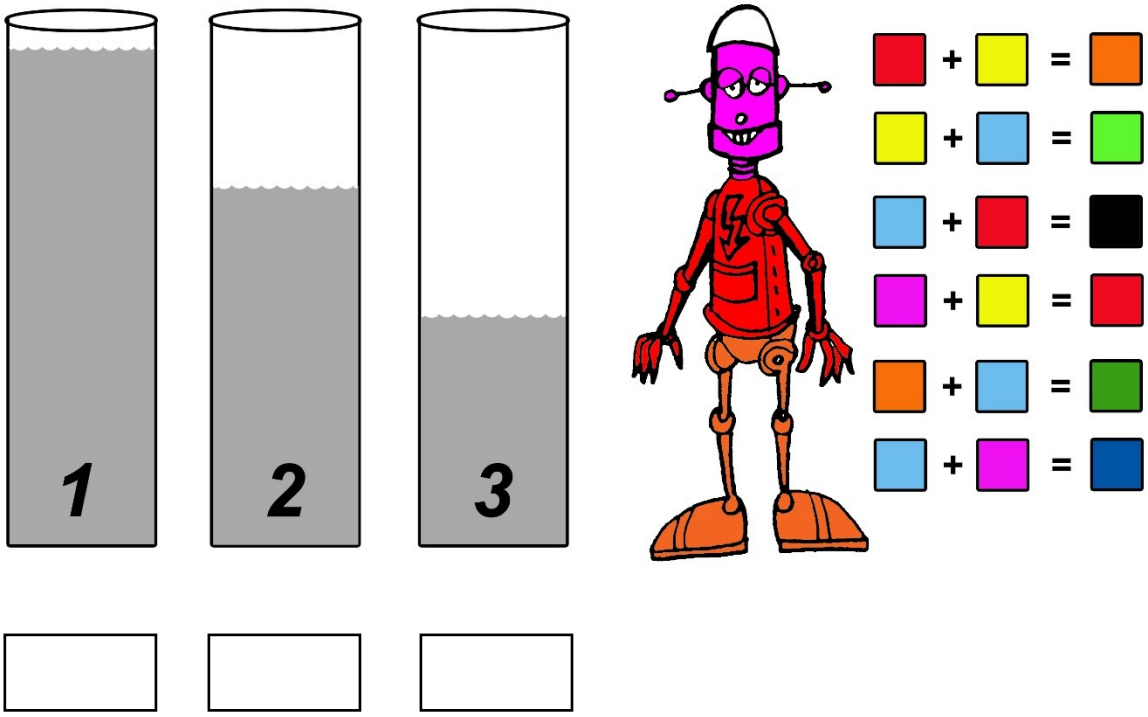


e)



Úloha č. 9 – Barvení robota DUDU

Bílý robot DUDU se chtěl obarvit, proto prošel postupně třemi nádržemi s barvou. První nádrž byla téměř plná, ve druhé byla barva přibližně do dvou třetin a ve třetí do výšky asi jedné třetiny. Protože barvy nestihly na robotovi zaschnout, smíchaly se a vznikly barvy nové. Například pokud by se nabarvil nejprve žlutou a hned potom modrou, vznikla by barva světle zelená. Jaké barvy vzniknou smícháním je vidět v tabulce.

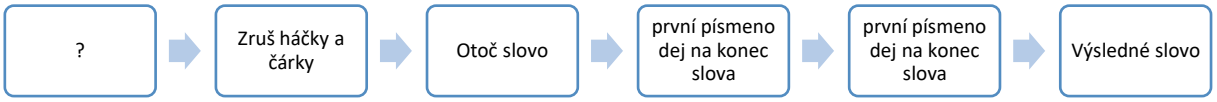


Jaké barvy byly v nádržích? Doplň do popisků jejich názvy.

Úloha č. 10 – Program na šifrování textu

Program na šifrování textu provádí následující kroky:

1. Otoč slovo
2. Zruš háčky a čárky
3. První písmeno dej na konec slova
4. První písmeno dej na konec slova
5. Napiš výsledné slovo



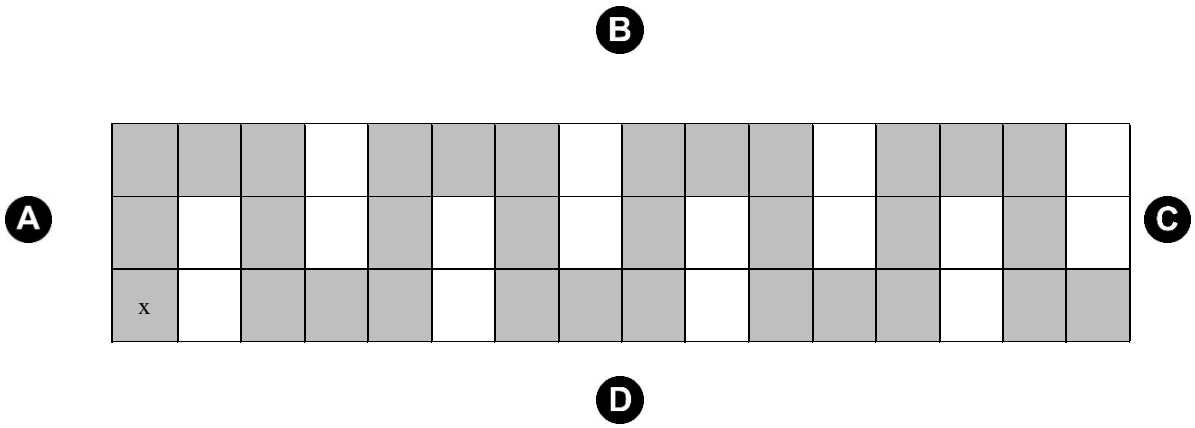
Příklad: SLON => SLON => NOLS => OLSN => LSNO

Jaké mohlo být původní slovo, které program zašifroval do shluku písmen ELPOS?

Odpověď: a) POPEL b) SOPEL c) POŠEL d) PLESO e) POSEL

Úloha č. 11 – Barvení dlaždic

Robot má za úkol natřít šedé dlaždice zeleně. Nachází se na místě označeném křížkem a natře každou dlaždici, kterou přejede. Rozumí následujícím pokynům A = jede jedno pole směrem A, B = jede jedno pole směrem B, C = jede jedno pole směrem C, D = jede jedno pole směrem D. Příkazy lze zkracovat pomocí čísel. Pokud robot dostane instrukci 3B, pojede tři pole směrem B. Nikdy se nesmí vrátit zpět na čerstvě natřené dlaždice. V jeho paměti však zůstalo jen velmi málo místa. Který

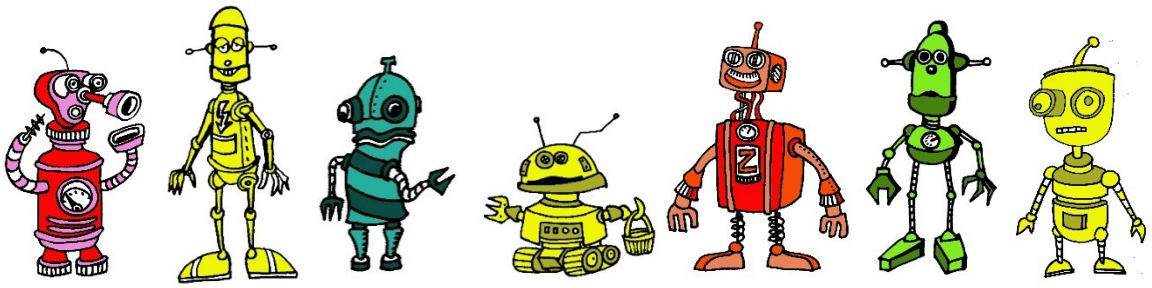


nejkratší program povede k natření šedých dlaždic na modro.

Odpověď

a) 2B 2C 2D 1C 2B 2C 2D 1C 2B 2C 2D 1C 2B 2C 2D 1C
b) 4x (2B 2C A 2C)
c) 4x (2B 2C 2D 2C)
d) 2x (2A 2B 2C 2D)
e) 2x (2B 2C 2D 2C 2B 2C 2D 2C)

Úloha č. 12 – Místnost plná robotů



V místnosti je sedm robotů, kteří poslouchají tvoje příkazy. Důležitější je pro ně vždy dřívější příkaz, který se jich týká. Dostali od tebe tyto tři příkazy:

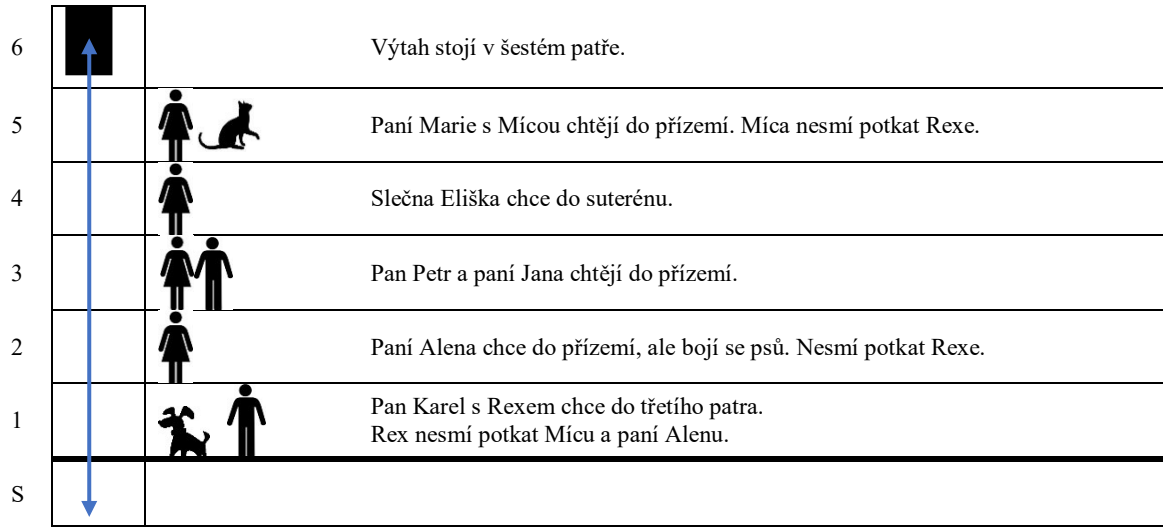
1) Všichni, kdo jste žlutí, dělejte opak dalších příkazů.
2) Kdo máte nohy, nechoďte ke mně.
3) Kdo máte pásy, přesuňte se ke mně.

Kolik robotů bude po těchto příkazech u tebe?

Odpověď: a) žádný b) dva d) čtyři e) šest f) všichni

Úloha č. 13 – Výtah

Pomocí programu ovládáš výtah, který na začátku stojí v šestém patře. Vyber program, který přepraví všechny čekající obyvatele domu tam, kam potřebují a současně se nepotkají pes a kočka, nebo Rex a paní Alena, která se psů bojí. Do výtahu se vejdou pouze tři lidé, nebo dva lidé a jedno zvíře. Pokud výtah v patře zastaví, nastoupí všichni, kdo mohou. Suterén je označen písmenem S, přízemí číslem 1.



Odpověď:

f) 3-2-1-5-1-S-1-3

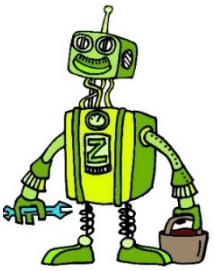
g) 5-1-3-4-2-1-S

h) 4-3-1-S-3-5-2-1

i) 4-3-S-1-2-5-1

j) 3-1-S-1-3-5-4-3

Úloha č. 14 – Cesta



Robot EGON dostal zadány do programu následující instrukce:

- Až v garáži vyměníš olej u tátova auta, zajdi vyzvednout ALENKU ze školky.
- Až se vrátíš domů, uvař polévku.
- Zajdi koupit do autoservisu olej do tátova auta.
- Než půjdeš koupit olej do auta kup po cestě v tržnici zeleninu na polévku.
- S ALENKOU jděte domů kolem pošty, kde vyzvedni balík.

V jakém pořadí EGON navštívil GARÁŽ, DOMOV, TRŽNICI, ŠKOLKU, AUTOSERVIS a POŠTU?

Vyber a zakroužkuj správnou odpověď.

Odpověď:

f) garáž > školka > pošta > domov > tržnice > autoservis

g) autoservis > tržnice > garáž > školka > pošta > domov

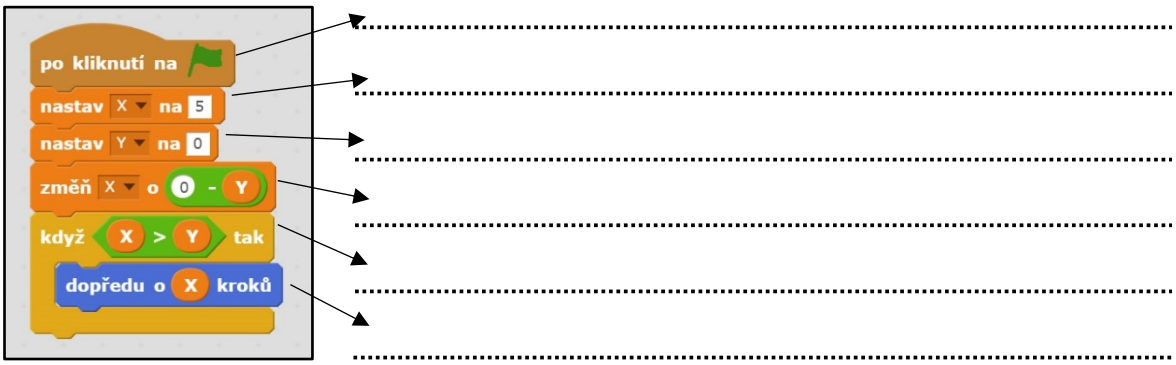
h) tržnice > autoservis > garáž > školka > pošta > domov

i) domov > tržnice > autoservis > školka > pošta > garáž

j) tržnice > garáž > autoservis > školka > pošta > domov

Úloha č. 15A a 15B – Dosazování

Kolik kroků dopředu udělá objekt po vykonání tohoto programu? Zakroužkuj správnou odpověď. Do řádků vedle obrázku popiš svými slovy, jak programu rozumíš. Každému bodu programu odpovídá jeden řádek popisu.



.....

.....

.....

.....

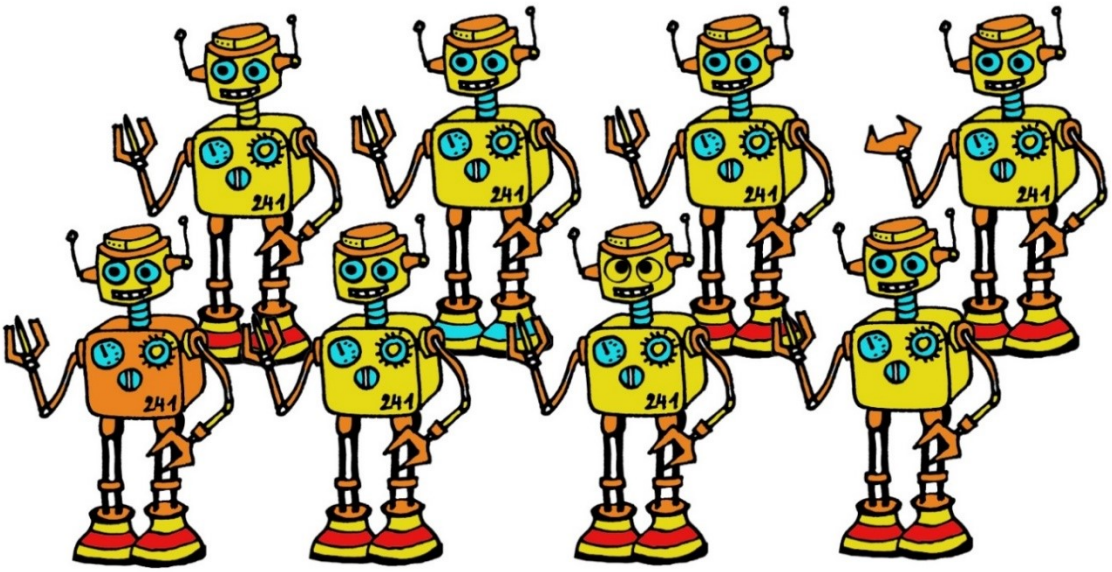
.....

.....

Odpověď: a) nula b) pět c) osm d) mínus tři e) jedenáct

Úloha č. 16 – Hra na schovávanou

Robot KAREL-VII se schoval mezi ostatními roboty ze stejné výrobní série. Má červené proužky na botách, modrý krk, dvě antény, tři prsty na pravé ruce, modré oči, žluté tělo a číslo 241 vepředu. Najdi robota BE-BE-TRI a zakroužkuj ho.



Příloha F – Seznam projektů

Třída	Skupina	Žák	Číslo projektu	Body Dr. Scratch	Hodnocení Dr. Scratch
5. třída	Skupina A	Žákyně 1	Projekt 1	14	Developing
		Žákyně 2	Projekt 2	12	Developing
		Žákyně 3	Projekt 3	15	Master
		Žák 4	Projekt 4	15	Master
		Žák 5	Projekt 5	16	Master
		Žák 6	Projekt 6	18	Master
		Žák 7	Projekt 7	17	Master
		Žák 8	Projekt 8	11	Developing
		Žák 9	Projekt 9	11	Developing
		Žák 10	Projekt 10	16	Master
		Žák 11	Projekt 11	13	Developing
	Skupina B	Žákyně 12	Projekt 12	17	Master
		Žákyně 13	Projekt 13	15	Master
		Žák 14	Projekt 14	8	Developing
		Žák 15	Projekt 15	12	Developing
		Žák 16	Projekt 16	0	Nelze hodnotit
		Žák 17	Projekt 17	10	Developing
		Žák 18	Projekt 18	8	Developing
		Žák 19	Projekt 19	15	Master
		Žák 20	Projekt 20	14	Developing
		Žák 21	Projekt 21	11	Developing
		Žák 22	Projekt 22	13	Developing
		Žák 23	Projekt 23	0	Nelze hodnotit
6. třída	Skupina C	Žákyně 24	Projekt 24	18	Master
		Žákyně 25	Projekt 25	10	Developing
		Žákyně 26	Projekt 26	13	Developing
		Žákyně 27	Projekt 27	7	Basic
		Žákyně 28	Projekt 28	13	Developing
		Žák 29	Projekt 29	15	Master
		Žák 30	Projekt 30	9	Developing
		Žák 31	Projekt 31	9	Developing
		Žák 32	Projekt 32	17	Master
		Žák 33	Projekt 33	14	Developing
		Žák 34	Projekt 34	17	Master
		Žák 35	Projekt 35	15	Master
		Žák 36	Projekt 36	12	Developing
	Skupina D	Žákyně 37	Projekt 37	14	Developing
		Žákyně 38	Projekt 38	11	Developing
		Žákyně 39	Projekt 39	14	Developing
		Žákyně 40	Projekt 40	13	Developing
		Žákyně 41	Projekt 41	10	Developing
		Žákyně 42	Projekt 42	9	Developing
		Žákyně 43	Projekt 43	0	Nelze hodnotit
		Žák 44	Projekt 44	13	Developing
		Žák 45	Projekt 45	7	Basic
		Žák 46	Projekt 46	10	Developing
		Žák 47	Projekt 47	12	Developing
		Žák 48	Projekt 48	16	Master

Příloha G – DOT 02 – Závěrečný žákovský dotazník

Třída: ..... Skupina: ..... chlapec ☐ dívka ☐

Zaškrtni políčko s variantou, která odpovídá skutečnosti.

	Ano	Ne
Založil/a sis účet na Scratch.mit.edu?		
Pracoval/a jsi na svém projektu i doma?		
Pracoval/a jsi ve Scratch i po ukončení výuky?		
Pracuješ ve Scratch i v současné době?		

Zaškrtni políčko s variantou, která nejlépe odpovídá tvému názoru na následující tvrzení.

	Rozhoně ano	Spíše ano	Spíše ne	Rozhodně ne	Nedovedu posoudit
Výuka programování byla zajímavá.					
Výuka ve Scratch byla hodně těžká.					
Naučil/a jsem se hodně o programování.					
Výuka ve Scratch mne bavila.					
Myslím, že kódům ve své hře rozumím.					
Bavila mě tvorba algoritmů.					
Bavilo mě programování hry.					
Bavilo mě vytváření grafiky pro hru.					
Bavilo mě hledání řešení.					
V práci se Scratch budu pokračovat.					
Všemu, co jsme se učili, jsem porozuměl/a.					
Chtěl/a bych se ve Scratch naučit ještě víc.					
Při výuce programování jsem se hodně naučil/a.					
S hrou co jsem tvořil/a jsem spokojený/á.					
Programování mě nezajímá.					
Chtěl/a bych umět vytvářet počítačové hry.					
Výuka ve Scratch byla pro mne snadná.					
Chtěl/a bych se stát programátorem.					
Předmět informatika je pro život potřebný.					
Umět programovat je pro život důležité.					
Algoritmické myšlení je pro můj život důležité.					
Logické myšlení je pro můj budoucí život důležité.					
Znalosti a dovednosti, které jsem při výuce získal, jsou pro život důležité.					

Kde nebo od koho jsi získal nejvíc svých počítačových dovedností?

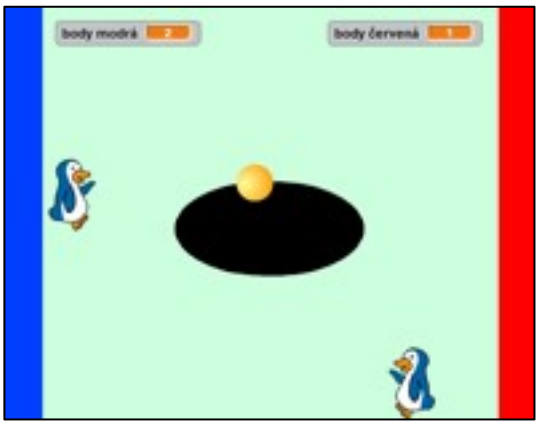
1. ..... 2. ..... 3. .....

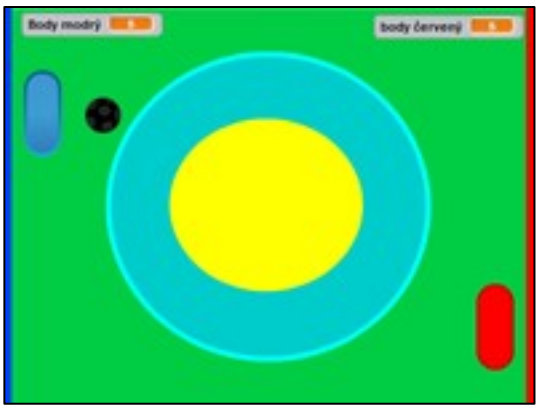
Napiš, co si myslíš o výuce programování ve Scratch. Jak tě bavila, co bys na výuce změnil/a, doplnil/a nebo naopak zrušil/a?

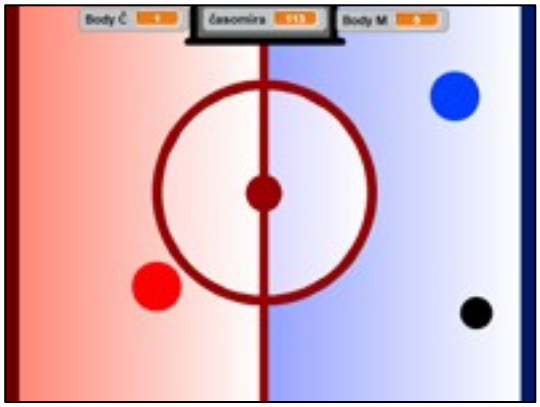
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Nakresli obrázek sebe při programování ve Scratch.

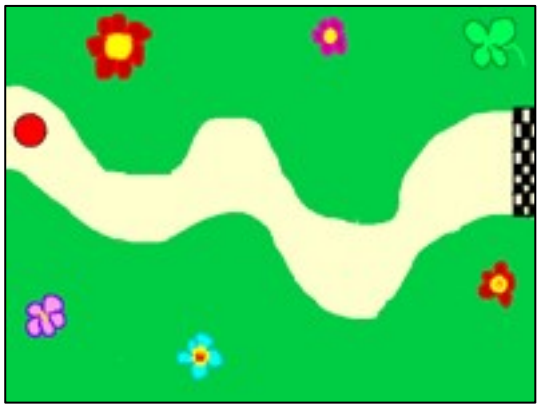
Příloha H – Ukázky dokončených her

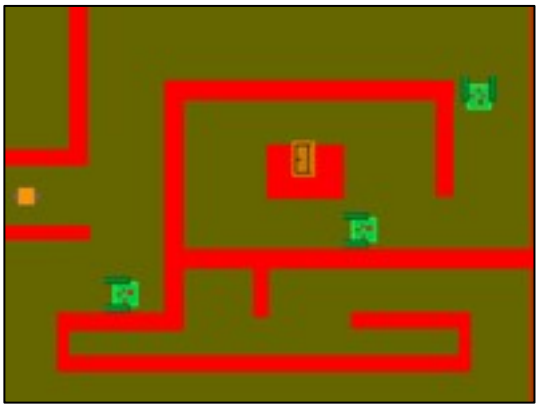


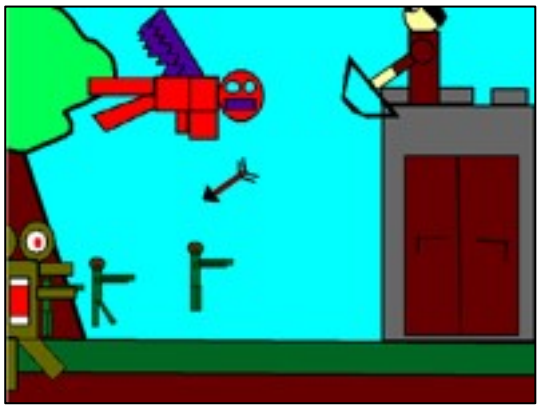


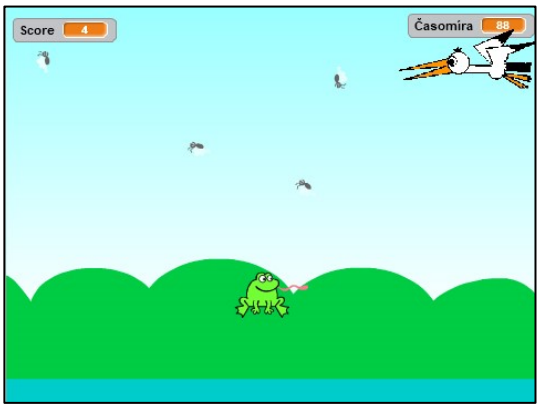

Body Č
Časomíra
Body M


body








Score
4
Časomíra
88

Příloha I – Vyplněný závěrečný žákovský dotazník (str. 2)

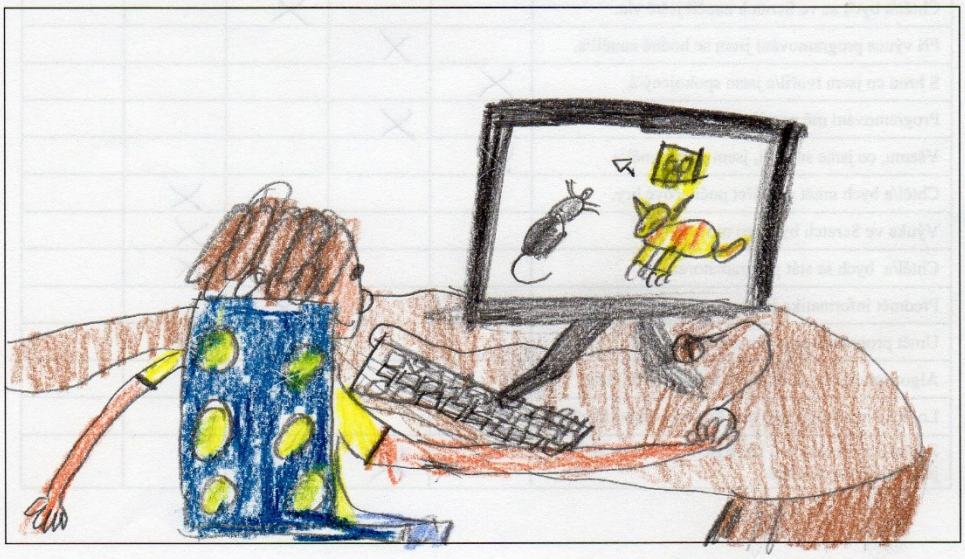
Kde nebo od koho jsi získal nejvíc svých počítačových dovedností?

1. ~~[illegible]~~ ve škole 2. od táty 3. od mamky

Napiš, co si myslíš o výuce programování ve Scratch. Jak tě bavila, co bys na výuce změnil/a, doplnil/a nebo naopak zrušil/a?

No, já jsem zpracování informatiky vůbec nerozumněl a spíš bych přešel se víc o programování, nevím by mne informatika nezajímala, že bych se víc věnoval všeobecný přehled ale mne se nezdá na programování nějaký problém. Podle mne není vůbec žádný problém v informatice ale nemám rád počítače. Doma žádný nemám. Takže jsem byl celkem v pohodě, myslel jsem si na to doma pořádně nepřipravil ale nejsem zavřený člověk. 1*

Nakresli obrázek sebe při programování ve Scratch.



BIBLIOGRAFICKÉ ÚDAJE

Autor: .....Bc. Milan Svoboda

Obor: .....Informační a komunikační technologie

Studijní program: .....Učitelství pro střední školy (N7504)

Studijní obor: .....N IT (7504T276)

Forma studia: .....Kombinovaná

Název práce: .....Rozvoj algoritmického myšlení žáků ZŠ ve výuce informaticky zaměřených předmětů s využitím SCRATCH

Rok: 2018

Počet stran textu bez příloh: 138

Celkový počet stran příloh: 50

Vedoucí práce: .....doc. RNDr. Miroslava Černochová, CSc.